目錄

Python 数据科学入门教程

> 来源：Data Analysis
>
> 译者：飞龙
>

- 在线阅读
- PDF格式
- EPUB格式
- MOBI格式
- 代码仓库

赞助我



Matplotlib 入门教程

第一章 Matplotlib 简介

> 原文：Introduction to Matplotlib and basic line
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

欢迎阅读 Python 3+ Matplotlib 系列教程。 在本系列中，我们将涉及 Matplotlib 数据可视化模块的多个方面。 Matplotlib 能够创建多数类型的图表，如条形图，散点图，条形图，饼图，堆叠图，3D 图和地图图表。

首先，为了实际使用 Matplotlib，我们需要安装它。

如果你安装了更高版本的 Python，你应该能够打开 `cmd.exe` 或终端，然后执行：

```
pip install matplotlib
```

注意：如果上面的较短命令不工作，你可能需要执行 `C:/Python34/Scripts/pip install matplotlib` 。

如果在导入 `matplotlib` 时，你会收到类似『无命名模块』和模块名称的错误，这意味着你还需要安装该模块。 一个常见的问题是缺少名为 `six` 的模块。 这意味着你需要使用 `pip` 安装 `six` 。

或者，你可以前往 Matplotlib.org 并通过访问下载页面下载适当的版本进行安装。 请记住，因为你的操作系统为 64 位，你不一定需要 64 位版本的 Python。 如果你不打算尝试 64 位，你可以使用 32 位。 打开 IDLE 并阅读顶部。 如果它说你是 64 位，你就是 64 位，如果它说是 32 位，那么你就是 32 位。 一旦你安装了 Python，你就做好了准备，你可以编写任何你想要的逻辑。 我喜欢使用 IDLE 来编程，但你可以随意使用任何你喜欢的东西。

```
import matplotlib.pyplot as plt
```

这一行导入集成的 `pyplot` ，我们将在整个系列中使用它。 我们将 `pyplot` 导入为 `plt` ，这是使用 `pylot` 的 python 程序的传统惯例。

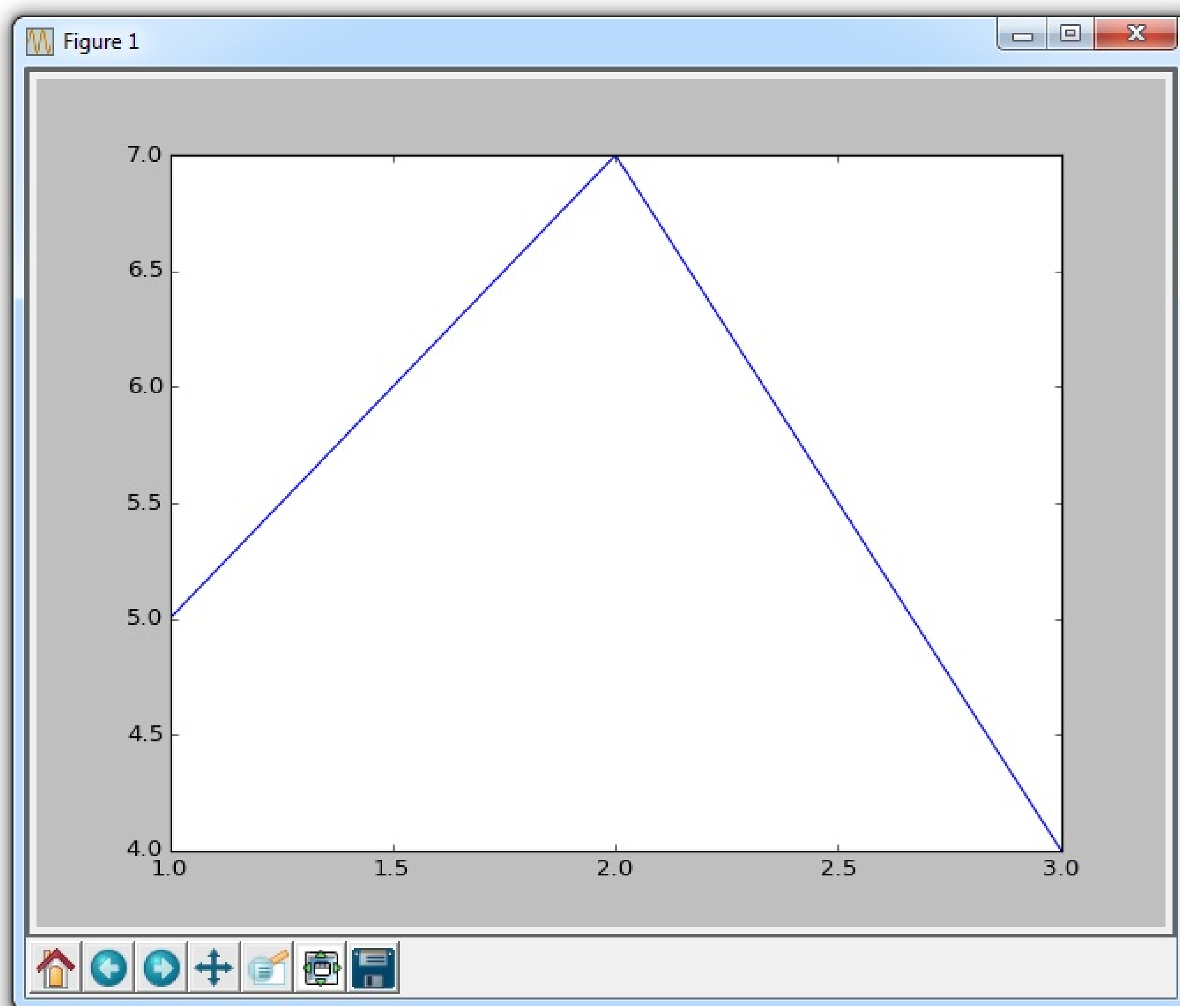
```
plt.plot([1,2,3],[5,7,4])
```

接下来，我们调用 `plot` 的 `.plot` 方法绘制一些坐标。 这个 `.plot` 需要许多参数，但前两个是 `'x'` 和 `'y'` 坐标，我们放入列表。 这意味着，根据这些列表我们拥有 3 个坐标： `1,5` `2,7` 和 `3,4` 。

`plt.plot` 在后台『绘制』这个绘图，但绘制了我们想要的一切之后，当我们准备好的时候，我们需要把它带到屏幕上。

```
plt.show()
```

这样，应该弹出一个图形。 如果没有，有时它可以弹出，或者你可能得到一个错误。 你的图表应如下所示：



这个窗口是一个 matplotlib 窗口，它允许我们查看我们的图形，以及与它进行交互和访问。 你可以将鼠标悬停在图表上，并查看通常在右下角的坐标。 你也可以使用按钮。 它们可能在不同的位置，但在上图中，这些按钮在左下角。

`Home`（主页）



一旦你开始浏览你的图表，主页按钮会帮助你。 如果你想要返回原始视图，可以单击它。 在浏览图表之前单击此按钮将不会生效。

`Forward/Back`（前进/后退）



这些按钮可以像浏览器中的前进和后退按钮一样使用。 你可以单击这些来移回到你之前的位置，或再次前进。

`Pan`（平移）



你可以点击平移按钮，之后点击并拖拽你的图表。

`Zoom`（缩放）

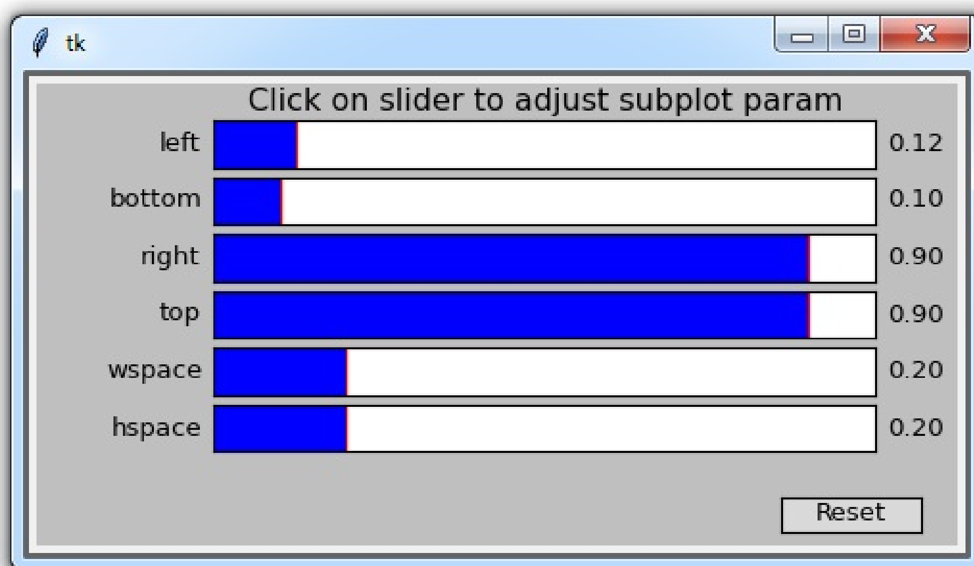


缩放按钮可让你单击它，然后单击并拖动出要放大的方形区域。 放大需要左键单击并拖动。 你也可以右键单击并拖动来缩小。

`Configure Subplots`（配置子图）

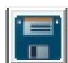


此按钮允许你对图形和绘图配置各种间距选项。 点击它会弹出：



每个蓝色条形都是一个滑块，它允许你调整内边距。 其中有些现在没有任何效果，因为没有任何其他子图。 前四个值调整图形到窗口边缘的边距。 之后 `wspace` 和 `hspace` 对应于当你绘制多个子图时，它们的水平或竖直间距。

`Save`（保存）



此按钮允许你以各种形式保存图形。

所以这是 matplotlib 的快速介绍，我们之后会涉及更多。

第二章 图例、标题和标签

> 原文：Legends, Titles, and Labels with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在本教程中，我们将讨论 Matplotlib 中的图例，标题和标签。 很多时候，图形可以不言自明，但是图形带有标题，轴域上的标签和图例，来解释每一行是什么非常必要。

> 注：轴域（`Axes`）即两条坐标轴围城的区域。

从这里开始：

```
import matplotlib.pyplot as plt

x = [1,2,3]
y = [5,7,4]

x2 = [1,2,3]
y2 = [10,14,12]
```

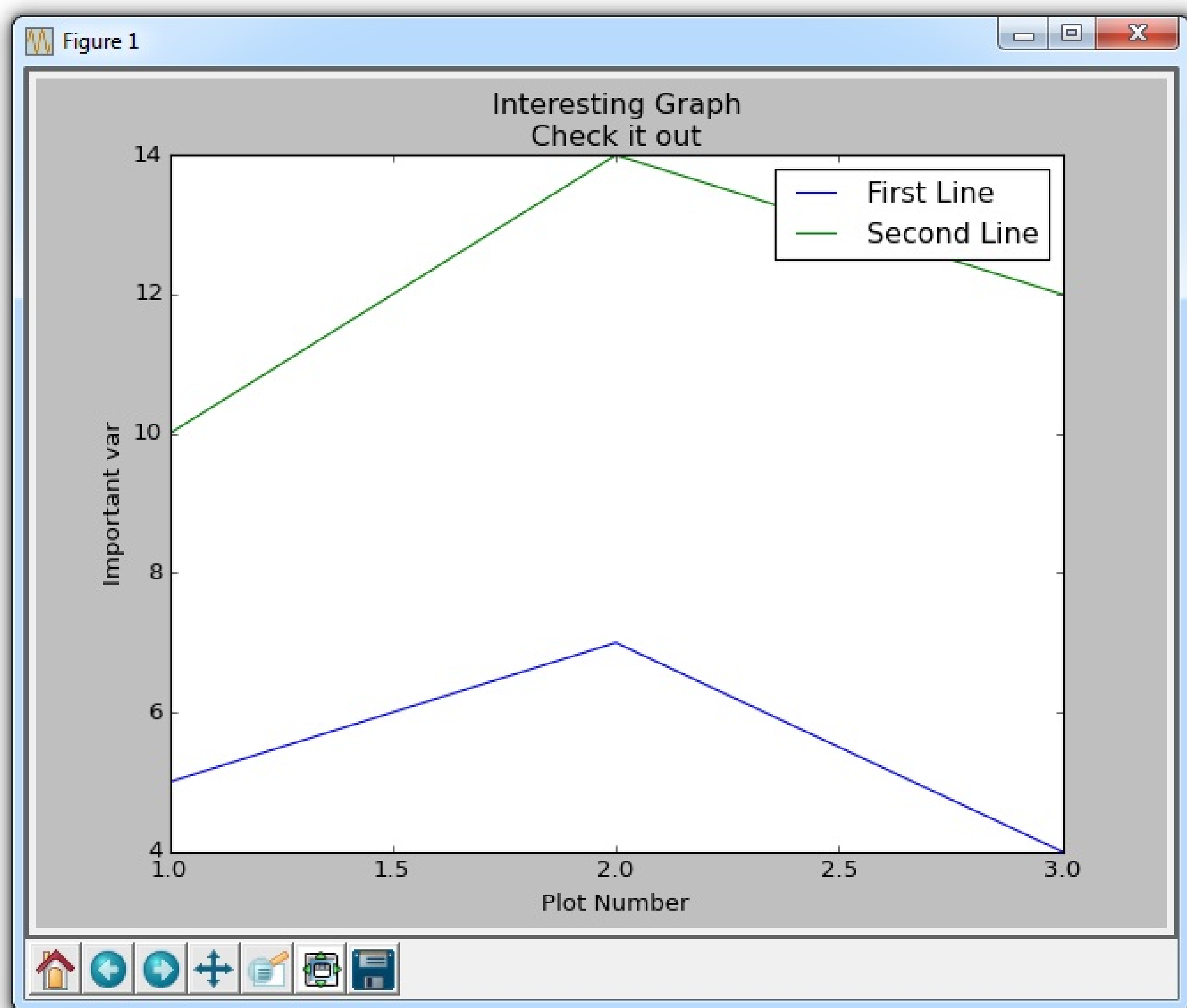
这样我们可以画出两个线条，接下来：

```
plt.plot(x, y, label='First Line')
plt.plot(x2, y2, label='Second Line')
```

在这里，我们绘制了我们已经看到的东西，但这次我们添加另一个参数`label`。这允许我们为线条指定名称，我们以后可以在图例中显示它。 我们的其余代码为：

```
plt.xlabel('Plot Number')
plt.ylabel('Important var')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```

使用`plt.xlabel`和`plt.ylabel`，我们可以为这些相应的轴创建标签。 接下来，我们可以使用`plt.title`创建图的标题，然后我们可以使用`plt.legend()`生成默认图例。 结果图如下：


Figure 1
Interesting Graph
Check it out
First Line
Second Line
Important var
Plot Number
14
12
10
8
6
4
1.0
1.5
2.0
2.5
3.0

第三章 条形图和直方图

> 原文：Bar Charts and Histograms with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

这个教程中我们会涉及条形图和直方图。我们先来看条形图：

```
import matplotlib.pyplot as plt

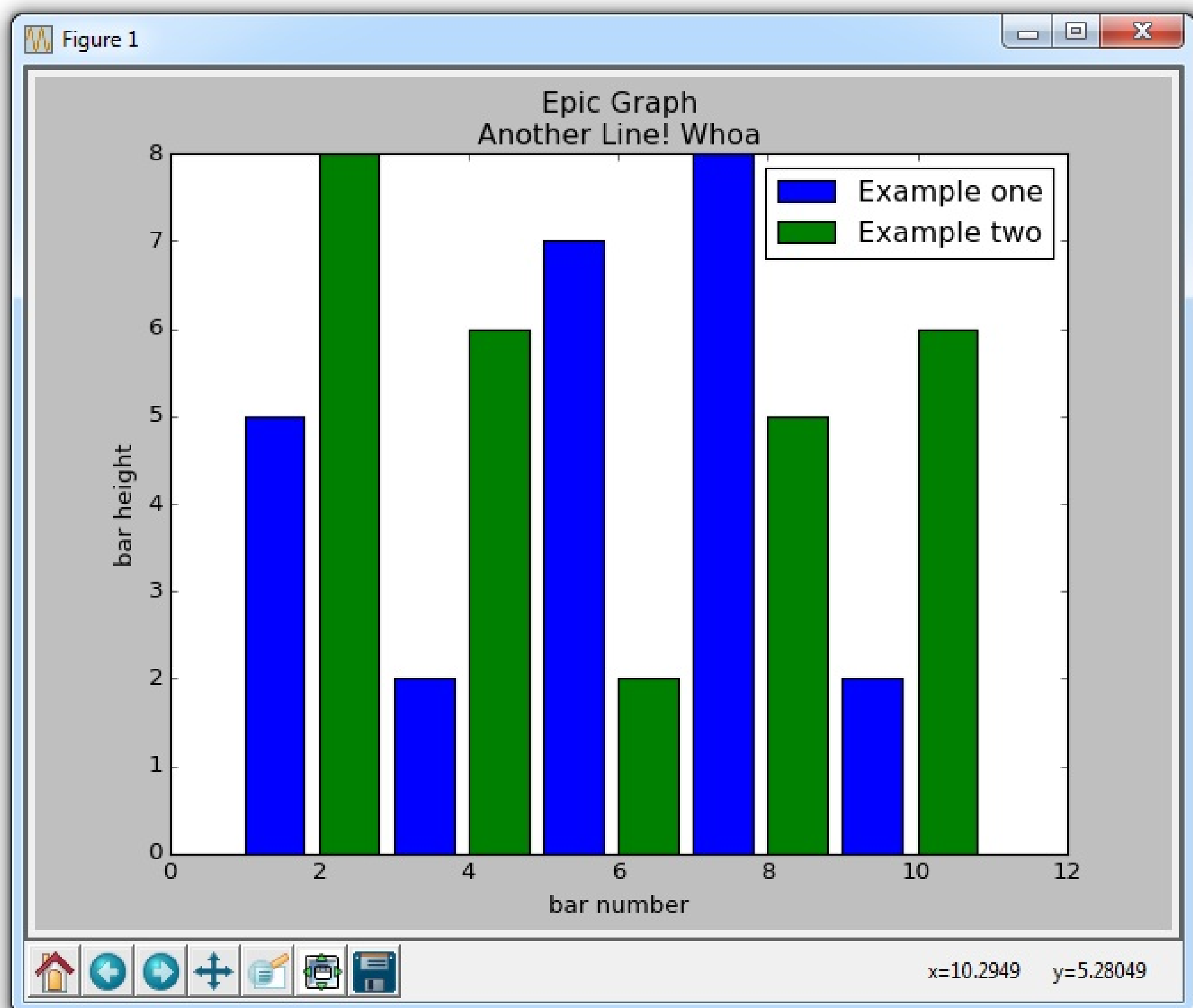
plt.bar([1,3,5,7,9],[5,2,7,8,2], label="Example one")

plt.bar([2,4,6,8,10],[8,6,2,5,6], label="Example two", color='g'
)
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')

plt.title('Epic Graph\nAnother Line! Whoa')

plt.show()
```

`plt.bar` 为我们创建条形图。如果你没有明确选择一种颜色，那么虽然做了多个图，所有的条看起来会一样。这让我们有机会使用一个新的 Matplotlib 自定义选项。你可以在任何类型的绘图中使用颜色，例如 `g` 为绿色，`b` 为蓝色，`r` 为红色，等等。你还可以使用十六进制颜色代码，如 `#191970` 。



接下来，我们会讲解直方图。 直方图非常像条形图，倾向于通过将区段组合在一起来显示分布。 这个例子可能是年龄的分组，或测试的分数。 我们并不是显示每一组的年龄，而是按照 20 ~ 25，25 ~ 30... 等等来显示年龄。 这里有一个例子：

```
import matplotlib.pyplot as plt

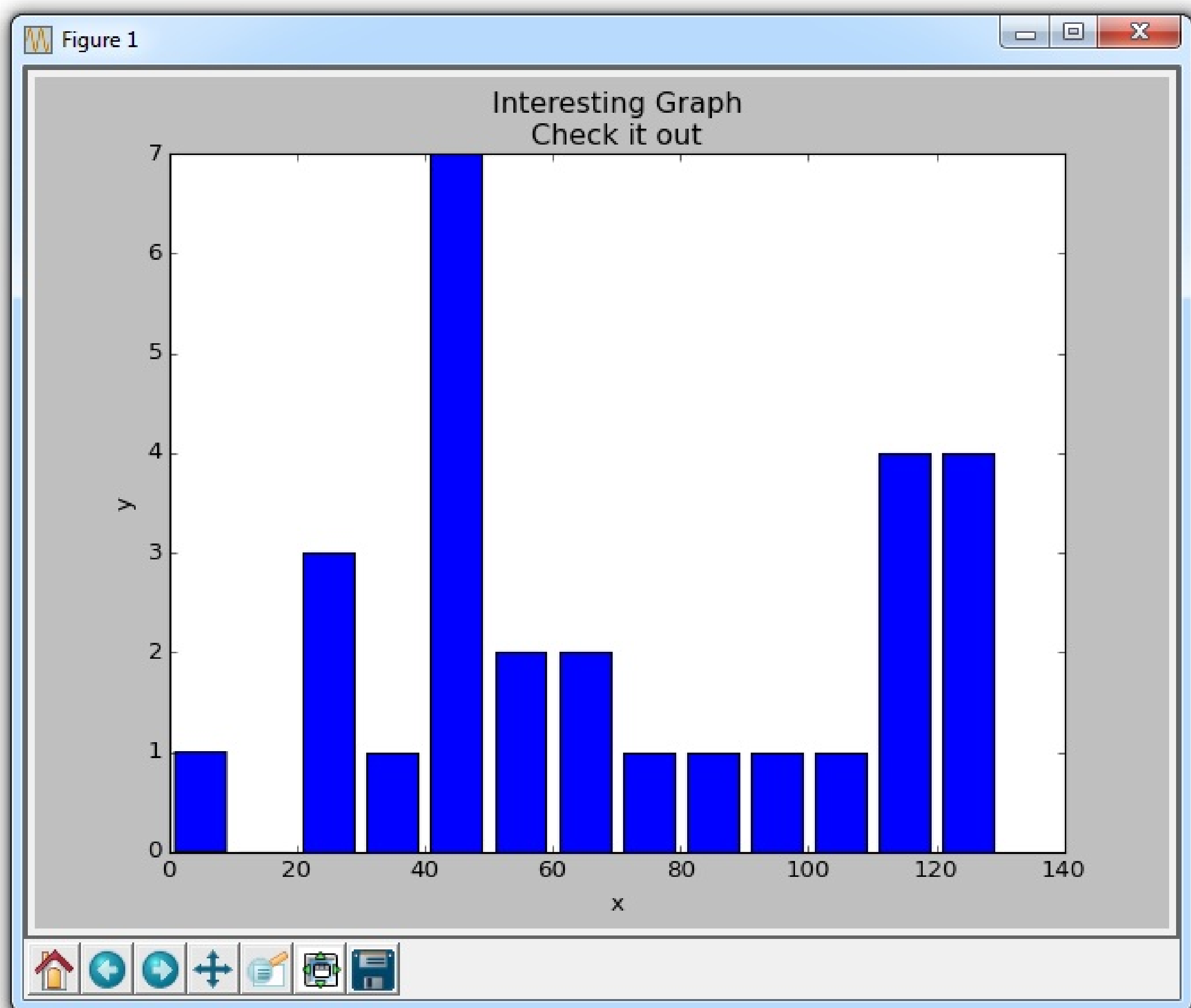
population_ages = [22,55,62,45,21,22,34,42,42,4,99,102,110,120,121,122,130,111,115,112,80,75,65,54,44,43,42,48]

bins = [0,10,20,30,40,50,60,70,80,90,100,110,120,130]

plt.hist(population_ages, bins, histtype='bar', rwidth=0.8)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```

产生的图表为：



对于 `plt.hist` ，你首先需要放入所有的值，然后指定放入哪个桶或容器。 在我们的例子中，我们绘制了一堆年龄，并希望以 10 年的增量来显示它们。 我们将条形的宽度设为 0.8，但是如果你想让条形变宽，或者变窄，你可以选择其他的宽度。

第四章 散点图

> 原文：Scatter Plots with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

接下来，我们将介绍散点图。散点图通常用于比较两个变量来寻找相关性或分组，如果你在 3 维绘制则是 3 个。

散点图的一些示例代码：

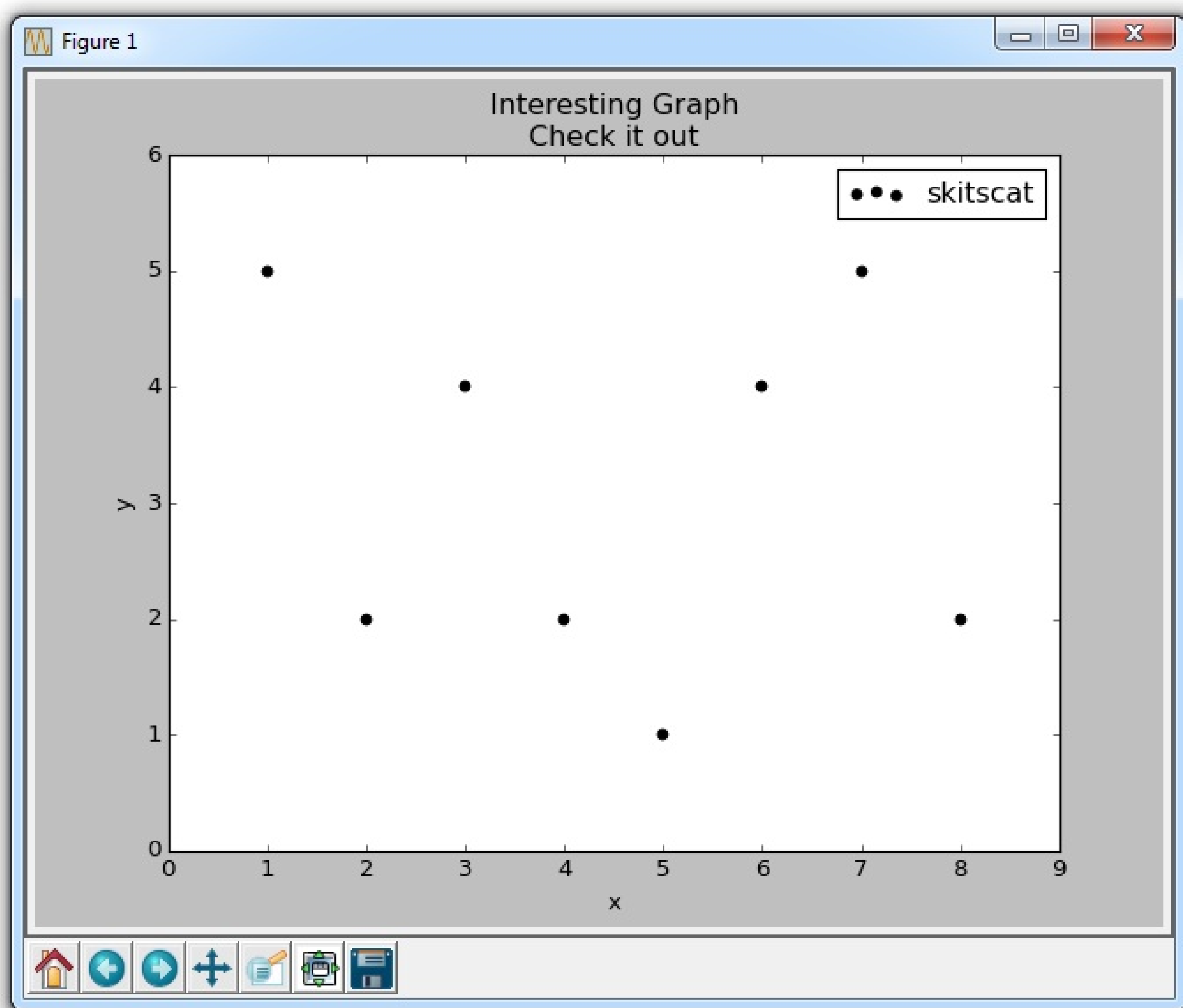
```
import matplotlib.pyplot as plt

x = [1,2,3,4,5,6,7,8]
y = [5,2,4,2,1,4,5,2]

plt.scatter(x,y, label='skitscat', color='k', s=25, marker="o")

plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```

结果为：



`plt.scatter` 不仅允许我们绘制 `x` 和 `y` ，而且还可以让我们决定所使用的标记颜色，大小和类型。 有一堆标记选项，请参阅 Matplotlib 标记文档中的所有选项。

第五章 堆叠图

> 原文：Stack Plots with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这篇 Matplotlib 数据可视化教程中，我们要介绍如何创建堆叠图。 堆叠图用于显示『部分对整体』随时间的关系。 堆叠图基本上类似于饼图，只是随时间而变化。

让我们考虑一个情况，我们一天有 24 小时，我们想看看我们如何花费时间。 我们将我们的活动分为：睡觉，吃饭，工作和玩耍。

我们假设我们要在 5 天的时间内跟踪它，因此我们的初始数据将如下所示：

```
import matplotlib.pyplot as plt

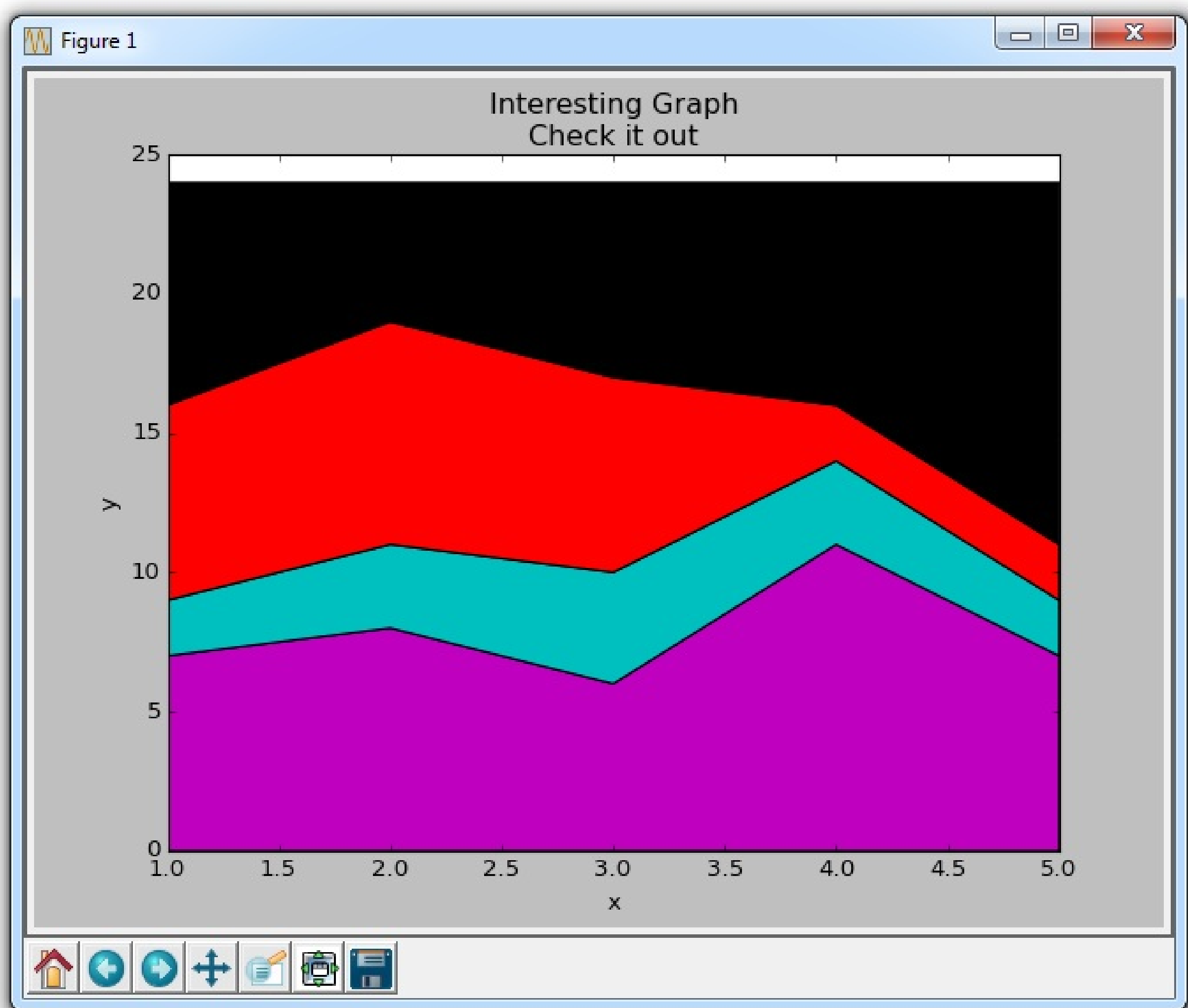
days = [1,2,3,4,5]

sleeping = [7,8,6,11,7]
eating =   [2,3,4,3,2]
working =  [7,8,7,2,2]
playing =  [8,5,7,8,13]
```

因此，我们的 `x` 轴将包括 `day` 变量，即 1, 2, 3, 4 和 5。然后，日期的各个成分保存在它们各自的活动中。 像这样绘制它们：

```
plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])

plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.show()
```



在这里，我们可以至少在颜色上看到，我们如何花费我们的时间。 问题是，如果不回头看代码，我们不知道什么颜色是什么。 下一个问题是，对于多边形来说，我们实际上不能为数据添加『标签』。 因此，在任何不止是线条，带有像这样的填充或堆叠图的地方，我们不能以固有方式标记出特定的部分。 这不应该阻止程序员。 我们可以解决这个问题：

```
import matplotlib.pyplot as plt

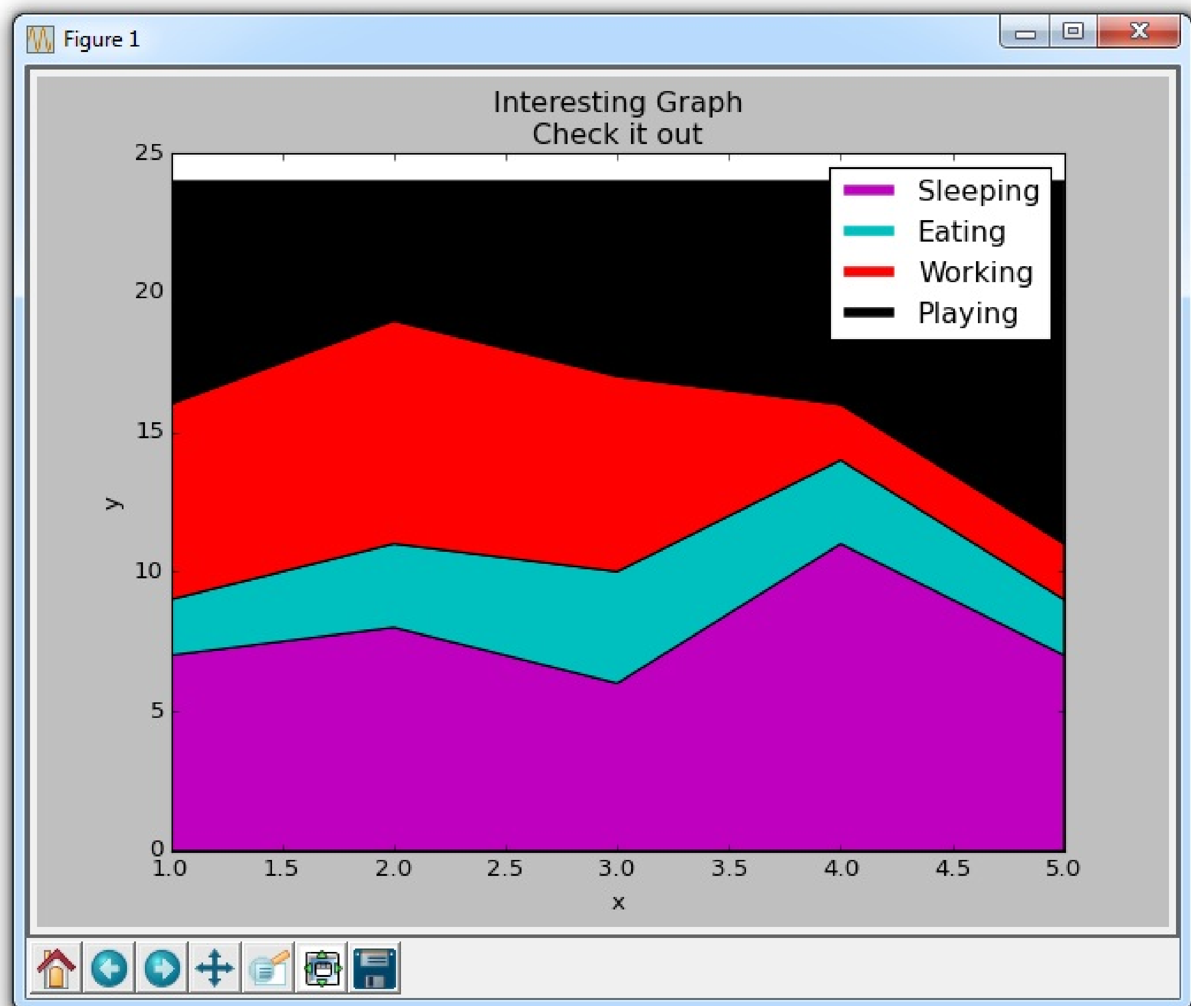
days = [1,2,3,4,5]

sleeping = [7,8,6,11,7]
eating =   [2,3,4,3,2]
working =  [7,8,7,2,2]
playing =  [8,5,7,8,13]


plt.plot([],[],color='m', label='Sleeping', linewidth=5)
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='k', label='Playing', linewidth=5)

plt.stackplot(days, sleeping,eating,working,playing, colors=['m',
'c','r','k'])

plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```



我们在这里做的是画一些空行，给予它们符合我们的堆叠图的相同颜色，和正确标签。 我们还使它们线宽为 5，使线条在图例中显得较宽。 现在，我们可以很容易地看到，我们如何花费我们的时间。

第六章 饼图

> 原文：Pie Charts with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

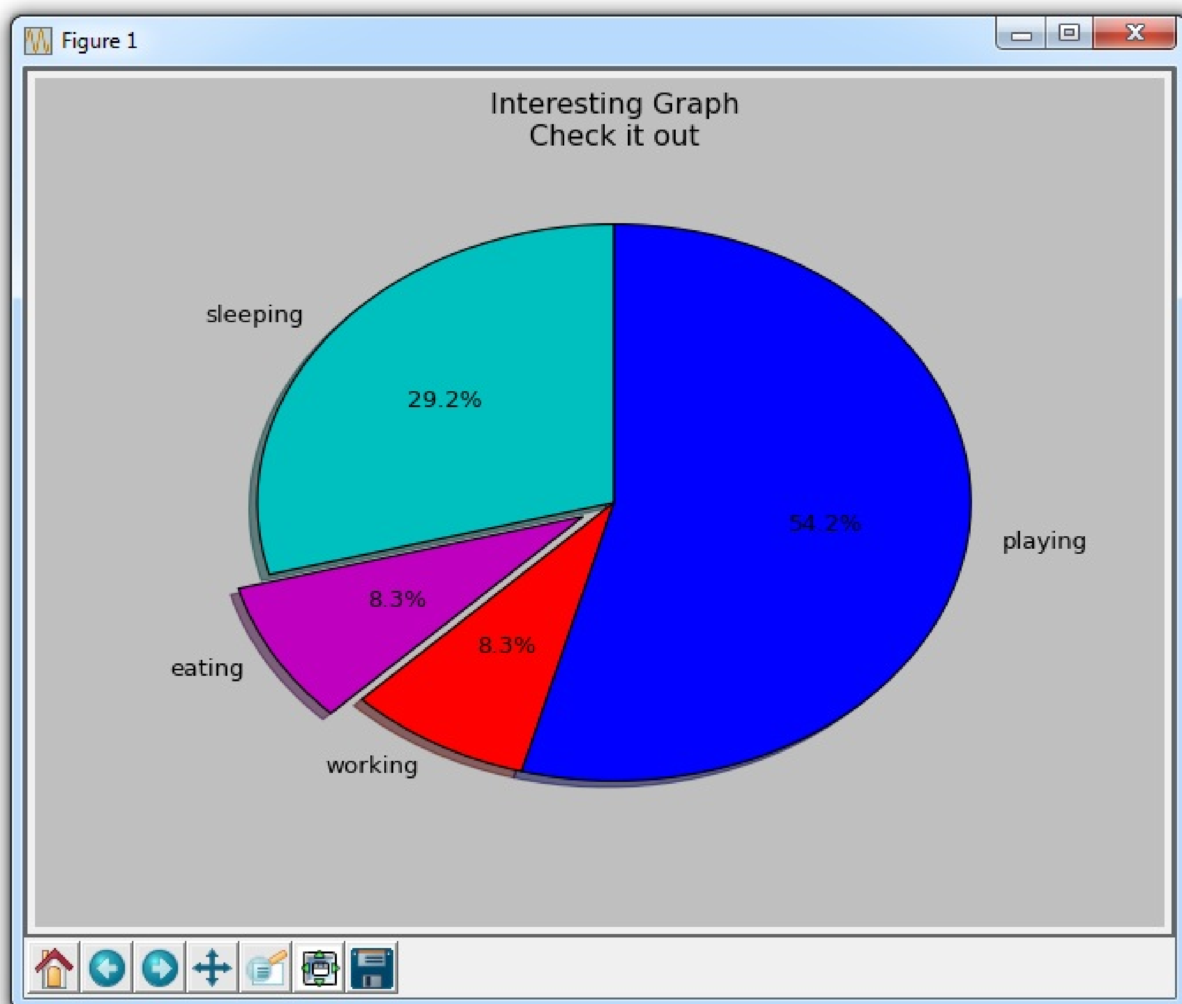
饼图很像堆叠图，只是它们位于某个时间点。 通常，饼图用于显示部分对于整体的情况，通常以%为单位。 幸运的是，Matplotlib 会处理切片大小以及一切事情，我们只需要提供数值。

```
import matplotlib.pyplot as plt

slices = [7,2,2,13]
activities = ['sleeping','eating','working','playing']
cols = ['c','m','r','b']

plt.pie(slices,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow= True,
        explode=(0,0.1,0,0),
        autopct='%1.1f%%')

plt.title('Interesting Graph\nCheck it out')
plt.show()
```



在 `plt.pie` 中，我们需要指定『切片』，这是每个部分的相对大小。 然后，我们指定相应切片的颜色列表。 接下来，我们可以选择指定图形的『起始角度』。 这使你可以在任何地方开始绘图。 在我们的例子中，我们为饼图选择了 90 度角，这意味着第一个部分是一个竖直线条。 接下来，我们可以选择给绘图添加一个字符大小的阴影，然后我们甚至可以使用 `explode` 拉出一个切片。

我们总共有四个切片，所以对于 `explode` ，如果我们不想拉出任何切片，我们传入 `0,0,0,0` 。 如果我们想要拉出第一个切片，我们传入 `0.1,0,0,0` 。

最后，我们使用 `autopct` ，选择将百分比放置到图表上面。

第七章 从文件加载数据

> 原文：Loading Data from Files for Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

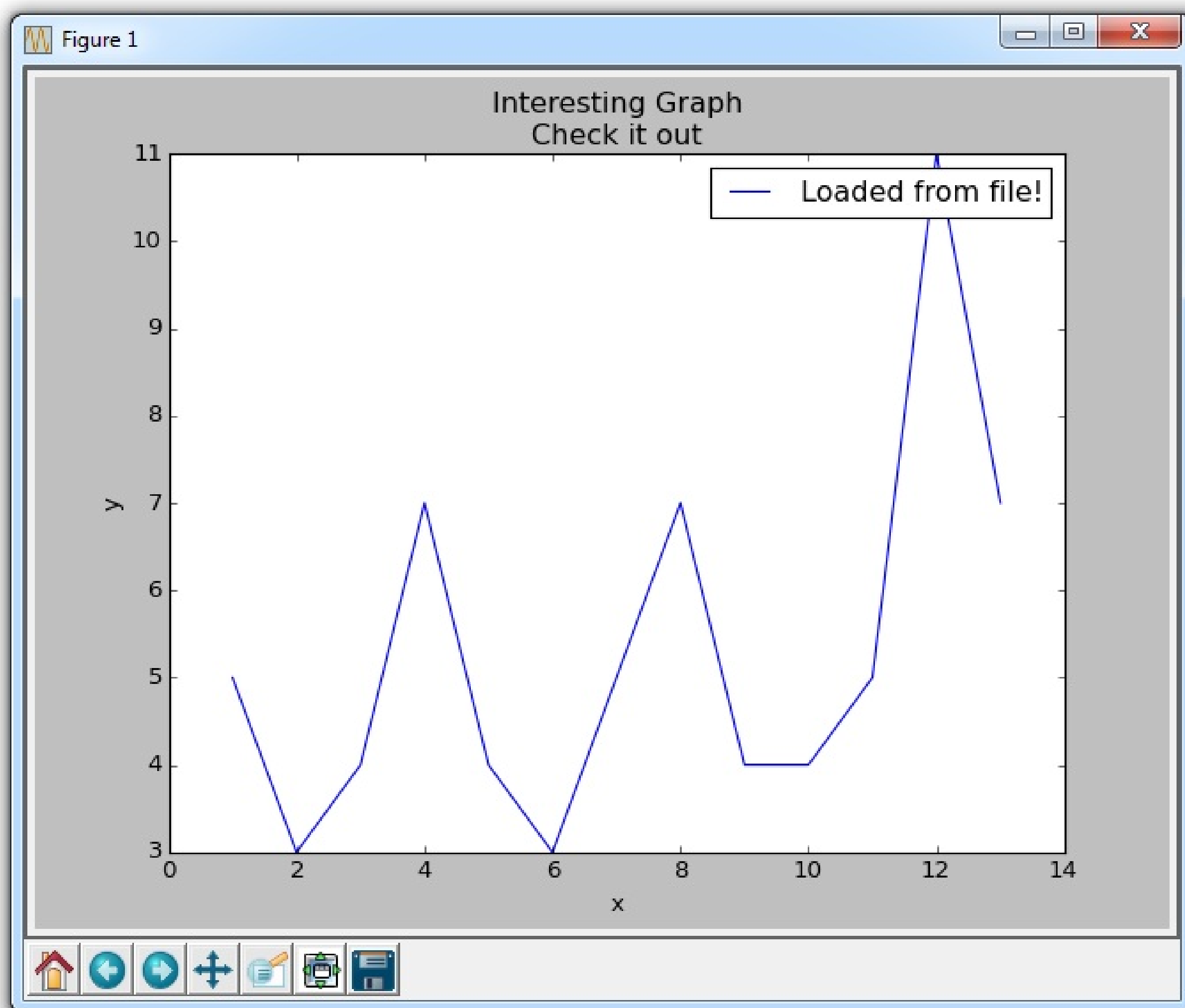
很多时候，我们想要绘制文件中的数据。 有许多类型的文件，以及许多方法，你可以使用它们从文件中提取数据来图形化。 在这里，我们将展示几种方法。 首先，我们将使用内置的 `csv` 模块加载CSV文件，然后我们将展示如何使用 NumPy（第三方模块）加载文件。

```
import matplotlib.pyplot as plt
import csv

x = []
y = []

with open('example.txt','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append(int(row[0]))
        y.append(int(row[1]))

plt.plot(x,y, label='Loaded from file!')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```



这里，我们打开样例文件，包含以下数据：

```
1,5
2,3
3,4
4,7
5,4
6,3
7,5
8,7
9,4
10,4
```

接下来，我们使用 `csv` 模块读取数据。 `csv` 读取器自动按行分割文件，然后使用我们选择的分隔符分割文件中的数据。 在我们的例子中，这是一个逗号。 注意： `csv` 模块和 `csv reader` 不需要文件在字面上是一个.csv文件。 它可以是任何具有分隔数据的简单的文本文件。

一旦我们这样做了，我们将索引为 0 的元素存储到 `x` 列表，将索引为 1 的元素存储到 `y` 列表中。 之后，我们都设置好了，准备绘图，然后显示数据。

虽然使用 CSV 模块是完全正常的，但使用 NumPy 模块来加载我们的文件和数据，可能对我们更有意义。 如果你没有 NumPy，你需要按下面的步骤来获取它。为了了解安装模块的更多信息，请参阅 pip 教程。 大多数人应该都能打开命令行，并执行 `pip install numpy` 。

如果不能，请参阅链接中的教程。

一旦你安装了 NumPy，你可以编写如下代码：

```
import matplotlib.pyplot as plt
import numpy as np

x, y = np.loadtxt('example.txt', delimiter=',', unpack=True)
plt.plot(x,y, label='Loaded from file!')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Interesting Graph\nCheck it out')
plt.legend()
plt.show()
```

结果应该是相同的图表。 稍后，当我们加载数据时，我们可以利用 NumPy 为我们做一些更多的工作，但这是教程未来的内容。 就像 `csv` 模块不需要一个特地的 `.csv` 一样， `loadtxt` 函数不要求文件是一个 `.txt` 文件，它可以是一个 `.csv` ，它甚至可以是一个 python 列表对象。

第八章 从网络加载数据

> 原文：Data from the Internet for Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

除了从文件加载数据，另一个流行的数据源是互联网。 我们可以用各种各样的方式从互联网加载数据，但对我们来说，我们只是简单地读取网站的源代码，然后通过简单的拆分来分离数据。

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import matplotlib.dates as mdates


def graph_data(stock):

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrument/1.0/'+stock+'/chartdata;type=quote;range=10y/csv'

    source_code = urllib.request.urlopen(stock_price_url).read().decode()

    stock_data = []
    split_source = source_code.split('\n')

    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line:
                stock_data.append(line)
```

这里有很多步骤。首先，我们看到 `import` 。 `pyplot` 像往常一样导入，然后导入了 `numpy` ，然后是用于访问互联网的 `urllib` ，然后导入了 `matplotlib.dates` 作为 `mdates` ，它对于将日期戳转换为 matplotlib 可以理解的日期很有用。

接下来，我们开始构建我们的 `graph_data` 函数。在这里，我们首先定义包含股票数据的网址。之后，我们写一些 `urllib` 代码来访问该 URL，然后使用 `.read` 读取源代码，之后我们继续解码该数据。如果你使用 Python 2，则不必使用 `decode` 。

然后，我们定义一个空列表，这是我们将要放置股票数据的地方，我们也开始使用 `split_source` 变量拆分数据，以换行符拆分。

现在，如果你去看源代码，用 `stock` 替换 URL 中的 `+stock+` ，像 AAPL 那样，你可以看到大多数页面数据确实是股票定价信息，但有一些头信息我们需要过滤掉。为此，我们使用一些基本的过滤，检查它们来确保每行有 6 个数据点，然后确保术语 `values` 不在行中。

现在，我们已经解析了数据，并做好了准备。我们将使用 NumPy：

```
date, closep, highp, lowp, openp, volume = np.loadtxt(stock_data
,
                                                      delimiter=
',',
                                                      unpack=True
,
                                                      # %Y = ful
l year. 2015
                                                      # %y = par
tial year 15
                                                      # %m = num
ber month
                                                      # %d = num
ber day
                                                      # %H = hou
rs
                                                      # %M = min
utes
                                                      # %S = sec
onds
                                                      # 12-06-20
14
                                                      # %m-%d-%Y
                                                      converters
={0: bytespdate2num('%Y%m%d')})
```

我们在这里所做的是，使用 `numpy` 的 `loadtxt` 函数，并将这六个元素解构到六个变量。 这里的第一个参数是 `stock_data` ，这是我们加载的数据。 然后，我们指定 `delimiter` （这里是逗号），然后我们指定我们确实想要在这里解包变量，不是一个变量，而是我们定义的这组变量。 最后，我们使用可选的 `converters` 参数来指定我们要转换的元素（ `0` ），以及我们打算要怎么做。 我们传递一个名为 `bytespdate2num` 的函数，它还不存在，但我们下面会编写它。

第九章 时间戳的转换

> 原文：Converting date stamps for Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

本教程的重点是将来自 Yahoo finance API 的日期转换为 Matplotlib 可理解的日期。 为了实现它，我们要写一个新的函数， `bytespdate2num` 。

```
def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter
```

此函数接受数据，基于编码来解码数据，然后返回它。

将此应用于我们的程序的其余部分：

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import matplotlib.dates as mdates

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=10y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)
```

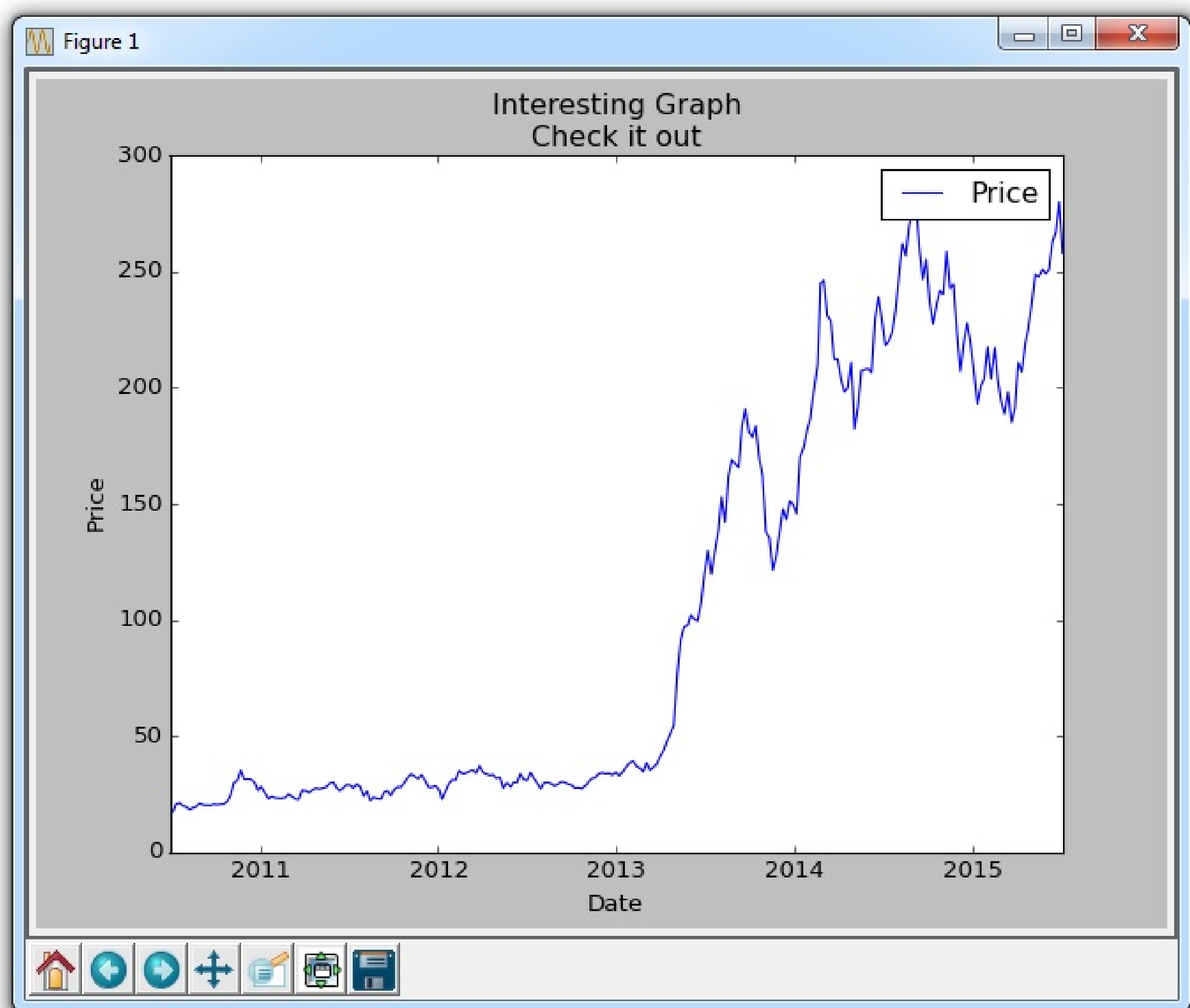
```
    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          # %Y =
 full year. 2015
                                                          # %y =
 partial year 15
                                                          # %m =
 number month
                                                          # %d =
 number day
                                                          # %H =
 hours
                                                          # %M =
 minutes
                                                          # %S =
 seconds
                                                          # 12-0
6-2014
                                                          # %m-%
d-%Y
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    plt.plot_date(date, closep,'-', label='Price')

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title('Interesting Graph\nCheck it out')
    plt.legend()
    plt.show()


graph_data('TSLA')
```

如果你绘制 TSLA，结果图应该看起来像这样：


Figure 1
Interesting Graph
Check it out
Price
300
250
200
150
100
50
0
2011
2012
2013
2014
2015
Date
Price

第十章 基本的自定义

> 原文：Basic customization with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在 Matplotlib 教程中，我们将讨论一些可能的图表自定义。 为了开始修改子图，我们必须定义它们。 我们很快会谈论他们，但有两种定义并构造子图的主要方法。现在，我们只使用其中一个，但我们会很快解释它们。

现在，修改我们的 `graph_data` 函数：

```
def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))
```

为了修改图表，我们需要引用它，所以我们将它存储到变量 `fig` 。 然后我们将 `ax1` 定义为图表上的子图。 我们在这里使用 `subplot2grid` ，这是获取子图的两种主要方法之一。 我们将深入讨论这些东西，但现在，你应该看到我们有 2 个元组，它们提供了 `(1,1)` 和 `(0,0)` 。 `1,1` 表明这是一个 1×1 网格。 然后 `0,0` 表明这个子图的『起点』将为 `0,0` 。

接下来，通过我们已经编写的代码来获取和解析数据：

```
stock_price_url = 'http://chartapi.finance.yahoo.com/instrument/
1.0/'+stock+'/chartdata;type=quote;range=10y/csv'
source_code = urllib.request.urlopen(stock_price_url).read().dec
ode()
stock_data = []
split_source = source_code.split('\n')
for line in split_source:
    split_line = line.split(',')
    if len(split_line) == 6:
        if 'values' not in line and 'labels' not in line:
            stock_data.append(line)

date, closep, highp, lowp, openp, volume = np.loadtxt(stock_data
,
                                                      delimiter=
',',
                                                      unpack=True
,
                                                      converters
={0: bytespdate2num('%Y%m%d')})
```

下面，我们这样绘制数据：

```
ax1.plot_date(date, closep,'-', label='Price')
```

现在，由于我们正在绘制日期，我们可能会发现，如果我们放大，日期会在水平方向上移动。但是，我们可以自定义这些刻度标签，像这样：

```
for label in ax1.xaxis.get_ticklabels():
    label.set_rotation(45)
```

这将使标签转动到对角线方向。 接下来，我们可以添加一个网格：

```
ax1.grid(True)
```

然后，其它东西我们保留默认，但我们也可能需要略微调整绘图，因为日期跑到了图表外面。 记不记得我们在第一篇教程中讨论的 `configure subplots` 按钮？ 我们不仅可以以这种方式配置图表，我们还可以在代码中配置它们，以下是我们设置这些参数的方式：

```
plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=0.90
, wspace=0.2, hspace=0)
```

现在，为了防止我们把你遗留在某个地方，这里是完整的代码：

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import matplotlib.dates as mdates

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=10y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)

    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

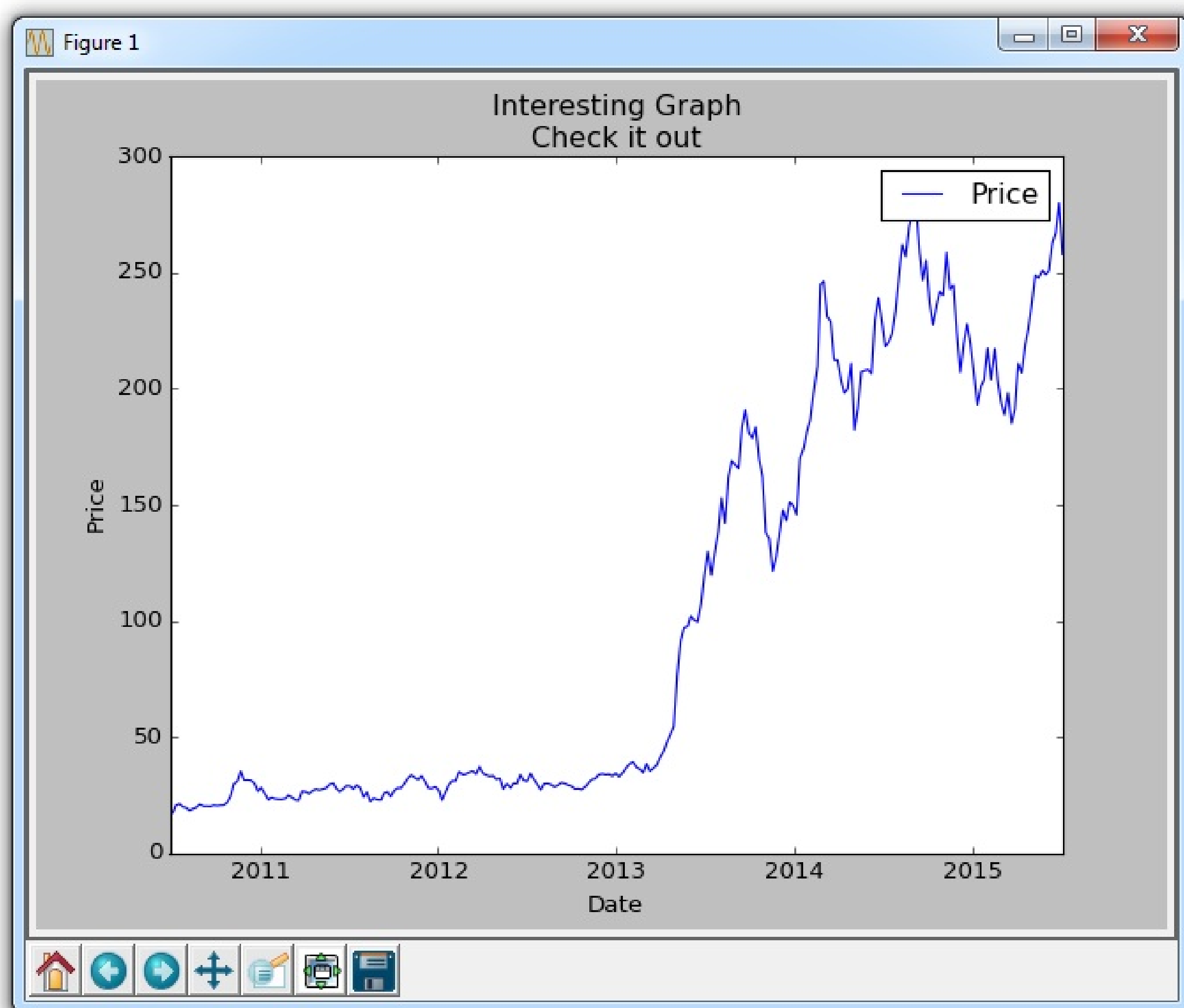
    ax1.plot_date(date, closep,'-', label='Price')
    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)
    ax1.grid(True)#, color='g', linestyle='-', linewidth=5)

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title('Interesting Graph\nCheck it out')
    plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
```

```
    plt.show()

graph_data('TSLA')
```

结果为：



第十一章 Unix 时间

> 原文：Unix Time with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们将介绍如何处理 unix 时间戳的转换，然后在图形中绘制日期戳。 使用 Yahoo Finance API，你会注意到，如果你使用较大的时间间隔，如 `1y` （一年），你会得到我们一直在使用的日期戳，但如果你使用 `10d` （10 天），反之你会得到 unix 时间的时间戳。

Unix 时间是 1970 年 1 月 1 日以后的秒数，它是跨程序的标准化时间表示方法。 也就是说，Matplotlib 并不欢迎 unix 时间戳。 这里是你可以使用 Matplotlib 来处理 unix 时间的方式：

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import datetime as dt
import matplotlib.dates as mdates

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=10d/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)

    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
```

```
data,
                                                          delimi
ter=',',
                                                          unpack=
True)
    dateconv = np.vectorize(dt.datetime.fromtimestamp)
    date = dateconv(date)

##    date, closep, highp, lowp, openp, volume = np.loadtxt(stoc
k_data,
##                                                          deli
miter=',',
##                                                          unpa
ck=True,
##                                                          conv
erters={0: bytespdate2num('%Y%m%d')})

    ax1.plot_date(date, closep,'-', label='Price')
    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)
    ax1.grid(True)#, color='g', linestyle='-', linewidth=5)

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title('Interesting Graph\nCheck it out')
    plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('TSLA')
```

所以在这里，我们所做的是 unix 时间的写入处理，并注释掉我们以前的代码，因为我们为之后的使用而保存它。 这里的主要区别是：

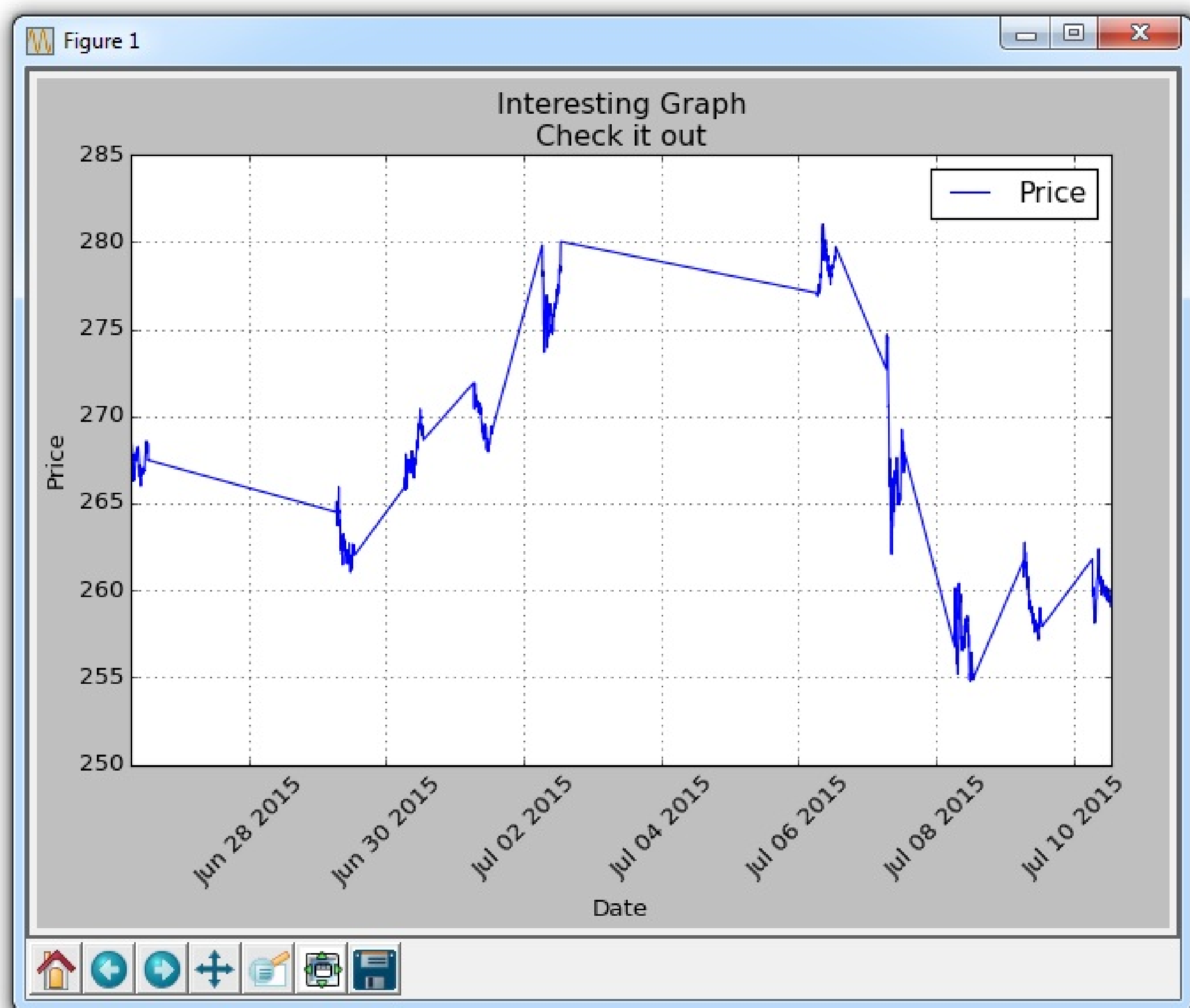
```
dateconv = np.vectorize(dt.datetime.fromtimestamp)
date = dateconv(date)
```

这里，我们将时间戳转换为日期戳，然后将其转换为 Matplotlib 想要的时间。

现在，由于某些原因，我的 unix 时间带有另一行包含 6 个元素的数据，并且它包含了术语 `label`，因此，在我们解析数据的 `for` 循环中，我们为你再添加一个需要注意的检查：

```
for line in split_source:
    split_line = line.split(',')
    if len(split_line) == 6:
        if 'values' not in line and 'labels' not in line:
            stock_data.append(line)
```

现在你的图表应该类似：



这里的所有扁平线条的原因是市场关闭。 有了这个短期数据，我们可以得到日内数据。 所以交易开放时有很多点，然后市场关闭时就没有了，然后又是一堆，然后又是没有。

第十二章 颜色和填充

> 原文：Colors and Fills with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在本教程中，我们将介绍一些更多的自定义，比如颜色和线条填充。

我们要做的第一个改动是将 `plt.title` 更改为 `stock` 变量。

```
plt.title(stock)
```

现在，让我们来介绍一下如何更改标签颜色。 我们可以通过修改我们的轴对象来实现：

```
ax1.xaxis.label.set_color('c')
ax1.yaxis.label.set_color('r')
```

如果我们运行它，我们会看到标签改变了颜色，就像在单词中那样。

接下来，我们可以为要显示的轴指定具体数字，而不是像这样的自动选择：

```
ax1.set_yticks([0,25,50,75])
```

接下来，我想介绍填充。 填充所做的事情，是在变量和你选择的一个数值之间填充颜色。 例如，我们可以这样：

```
ax1.fill_between(date, 0, closep)
```

所以到这里，我们的代码为：

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import datetime as dt
import matplotlib.dates as mdates

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
```

```
            return strconverter(s)
        return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=10y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

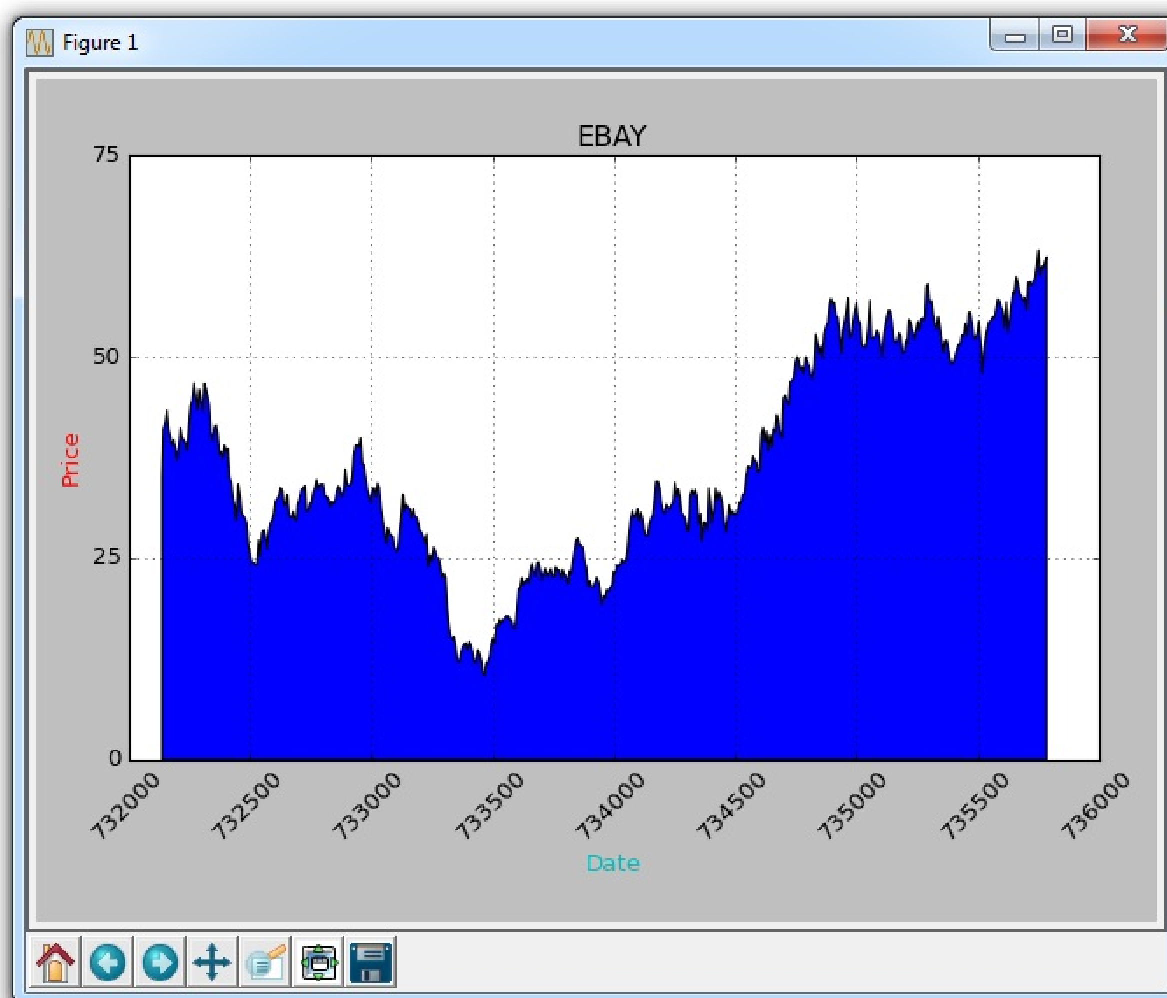
    ax1.fill_between(date, 0, closep)

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)
    ax1.grid(True)#, color='g', linestyle='-', linewidth=5)
    ax1.xaxis.label.set_color('c')
    ax1.yaxis.label.set_color('r')
    ax1.set_yticks([0,25,50,75])

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

结果为：



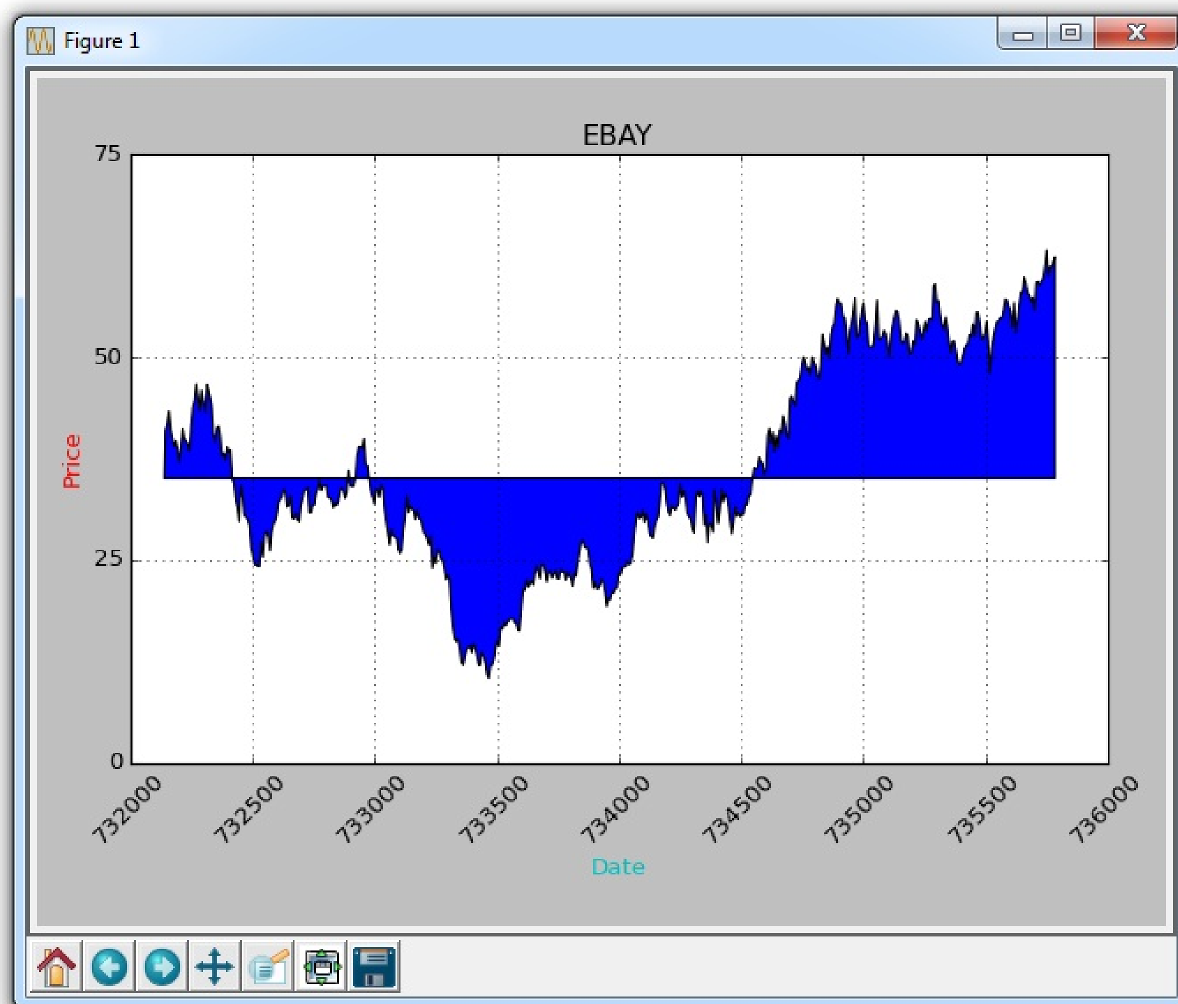
填充的一个问题是，我们可能最后会把东西都覆盖起来。 我们可以用透明度来解决它：

```
ax1.fill_between(date, 0, closep)
```

现在，让我们介绍条件填充。 让我们假设图表的起始位置是我们开始买入 eBay 的地方。 这里，如果价格低于这个价格，我们可以向上填充到原来的价格，然后如果它超过了原始价格，我们可以向下填充。 我们可以这样做：

```
ax1.fill_between(date, closep[0], closep)
```

会生成：



如果我们想用红色和绿色填充来展示收益/损失，那么与原始价格相比，绿色填充用于上升（注：国外股市的颜色和国内相反），红色填充用于下跌？没问题！我们可以添加一个 `where` 参数，如下所示：

```
ax1.fill_between(date, closep, closep[0],where=(closep > closep[0]), facecolor='g', alpha=0.5)
```

这里，我们填充当前价格和原始价格之间的区域，其中当前价格高于原始价格。我们给予它绿色的前景色，因为这是一个上升，而且我们使用微小的透明度。

我们仍然不能对多边形数据（如填充）应用标签，但我们可以像以前一样实现空线条，因此我们可以：

```
ax1.plot([],[],linewidth=5, label='loss', color='r',alpha=0.5)
ax1.plot([],[],linewidth=5, label='gain', color='g',alpha=0.5)

ax1.fill_between(date, closep, closep[0],where=(closep > closep[0
]), facecolor='g', alpha=0.5)
ax1.fill_between(date, closep, closep[0],where=(closep < closep[0
]), facecolor='r', alpha=0.5)
```

这向我们提供了一些填充，以及用于处理标签的空线条，我们打算将其显示在图例中。这里完整的代码是：

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import datetime as dt
import matplotlib.dates as mdates

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=10y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)

    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})
```

```
    ax1.plot_date(date, closep,'-', label='Price')

    ax1.plot([],[],linewidth=5, label='loss', color='r',alpha=0.5
)
    ax1.plot([],[],linewidth=5, label='gain', color='g',alpha=0.5
)

    ax1.fill_between(date, closep, closep[0],where=(closep > clo
sep[0]), facecolor='g', alpha=0.5)
    ax1.fill_between(date, closep, closep[0],where=(closep < clo
sep[0]), facecolor='r', alpha=0.5)

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)
    ax1.grid(True)#, color='g', linestyle='-', linewidth=5)
    ax1.xaxis.label.set_color('c')
    ax1.yaxis.label.set_color('r')
    ax1.set_yticks([0,25,50,75])

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

现在我们的结果是：


Figure 1
EBAY
75
50
25
0
Price
Price
loss
gain
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
Date

第十三章 边框和水平线条

> 原文：Spines and Horizontal Lines with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

欢迎阅读另一个定制教程，在这里我们使用 Matplotlib 讨论边框和水平线条。 有时候你可能想做的事情是改变边框的颜色，或者甚至完全删除它们。

图形的边框基本上是图形的边界，其中有刻度线等东西。为了改变边框的颜色，你可以做一些类似这样的事情：

```
ax1.spines['left'].set_color('c')
```

在这里，我们引用了我们的边框字典，表示我们要调整左边框，然后我们使用 `set_color` 方法将颜色设置为 `'c'` ，它是青色。

如果我们想删除所有边框怎么办？ 我们可以这样做：

```
ax1.spines['right'].set_visible(False)
ax1.spines['top'].set_visible(False)
```

这是非常类似的代码，删除了右边框和上边框。

很难看到我们修改了左边框的颜色，所以让我们通过修改线宽来使它变得很明显：

```
ax1.spines['left'].set_linewidth(5)
```

现在，左边框变成了非常粗也非常显眼的青色。 最后，如果我们想修改刻度参数怎么办？ 假如不想要黑色的日期，我们想要一些橙色的日期？ 没问题！

```
ax1.tick_params(axis='x', colors='#f06215')
```

现在我们的日期是橙色了！ 接下来，让我们来看看我们如何绘制一条水平线。 你当然可以将你创建的一组新数据绘制成一条水平线，但你不需要这样做。 你可以：

```
ax1.axhline(closep[0], color='k', linewidth=5)
```

所以在这里，我们的整个代码是：

```
import matplotlib.pyplot as plt
import numpy as np
import urllib
import datetime as dt
import matplotlib.dates as mdates

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=10y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    ax1.plot_date(date, closep,'-', label='Price')
    ax1.plot([],[],linewidth=5, label='loss', color='r',alpha=0.5
)
    ax1.plot([],[],linewidth=5, label='gain', color='g',alpha=0.5
)
    ax1.axhline(closep[0], color='k', linewidth=5)
    ax1.fill_between(date, closep, closep[0],where=(closep > clo
sep[0]), facecolor='g', alpha=0.5)
    ax1.fill_between(date, closep, closep[0],where=(closep < clo
sep[0]), facecolor='r', alpha=0.5)
```

```
    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)
    ax1.grid(True)
    #ax1.xaxis.label.set_color('c')
    #ax1.yaxis.label.set_color('r')
    ax1.set_yticks([0,25,50,75])

    ax1.spines['left'].set_color('c')
    ax1.spines['right'].set_visible(False)
    ax1.spines['top'].set_visible(False)
    ax1.spines['left'].set_linewidth(5)

    ax1.tick_params(axis='x', colors='#f06215')




    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('ebay')
```

结果为：


Figure 1
ebay
75
50
25
0
Price
Price
loss
gain
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
Date

第十四章 OHLC K 线图

> 原文：Candlestick OHLC graphs with Matplotlib
>
> 译者：飞龙
>
>

在 Matplotlib 教程中，我们将介绍如何在 Matplotlib 中创建开，高，低，关（OHLC）的 K 线图。 这些图表用于以精简形式显示时间序列股价信息。 为了实现它，我们首先需要导入一些模块：

```
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
```

我们引入了 `ticker`，允许我们修改图表底部的 `ticker` 信息。 然后我们从 `matplotlib.finance` 模块中引入 `candlestick_ohlc` 功能。

现在，我们需要组织我们的数据来和 matplotlib 协作。 如果你刚刚加入我们，我们得到的数据如下：

```
stock_price_url = 'http://chartapi.finance.yahoo.com/instrument/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
source_code = urllib.request.urlopen(stock_price_url).read().decode()
stock_data = []
split_source = source_code.split('\n')
for line in split_source:
    split_line = line.split(',')
    if len(split_line) == 6:
        if 'values' not in line and 'labels' not in line:
            stock_data.append(line)


date, closep, highp, lowp, openp, volume = np.loadtxt(stock_data,
                                                      delimiter=',',
                                                      unpack=True,
                                                      converters={0: bytespdate2num('%Y%m%d')})
```

现在，我们需要构建一个 Python 列表，其中每个元素都是数据。 我们可以修改我们的 `loadtxt` 函数，使其不解构，但随后我们还是希望引用特定的数据点。 我们可以解决这个问题，但是我们最后可能只拥有两个单独的数据集。 为此，我们执行

以下操作：

```
x = 0
y = len(date)
ohlc = []

while x < y:
    append_me = date[x], openp[x], highp[x], lowp[x], closep[x],
 volume[x]
    ohlc.append(append_me)
    x+=1
```

有了这个，我们现在将 OHLC 数据列表存储到我们的变量 `ohlc` 。现在我们可以这样绘制：

```
candlestick_ohlc(ax1, ohlc)
```

图表应该是这样：



不幸的是，`x` 轴上的 `datetime` 数据不是日期戳的形式。 我们可以处理它：

```
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
```

此外，红/黑着色依我看不是最好的选择。 我们应该使用绿色表示上升和红色表示下降。 为此，我们可以：

```
candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', colord
own='#db3f3f')
```

最后，我们可以将 `x` 标签设置为我们想要的数量，像这样：

```
ax1.xaxis.set_major_locator(mticker.MaxNLocator(10))
```

现在，完整代码现在是这样：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc

import numpy as np
import urllib
import datetime as dt


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
```

```
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1


    candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax1.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax1.grid(True)


    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

结果为：



还要注意，我们从前面的教程中删除了大部分 `ax1` 的修改。

第十五章 样式

> 原文：Styles with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们将讨论样式。 我们用于 Matplotlib 的样式非常相似于用于 HTML 页面的 CSS（层叠样式表）。 正如你在这里可以看到的，我们对图形所做的所有修改都会叠加，而且我们目前只有一个轴域。 我们可以使用 `for` 循环，至少使代码量降低，但我们也可以在 Matplotlib 中利用这些样式。

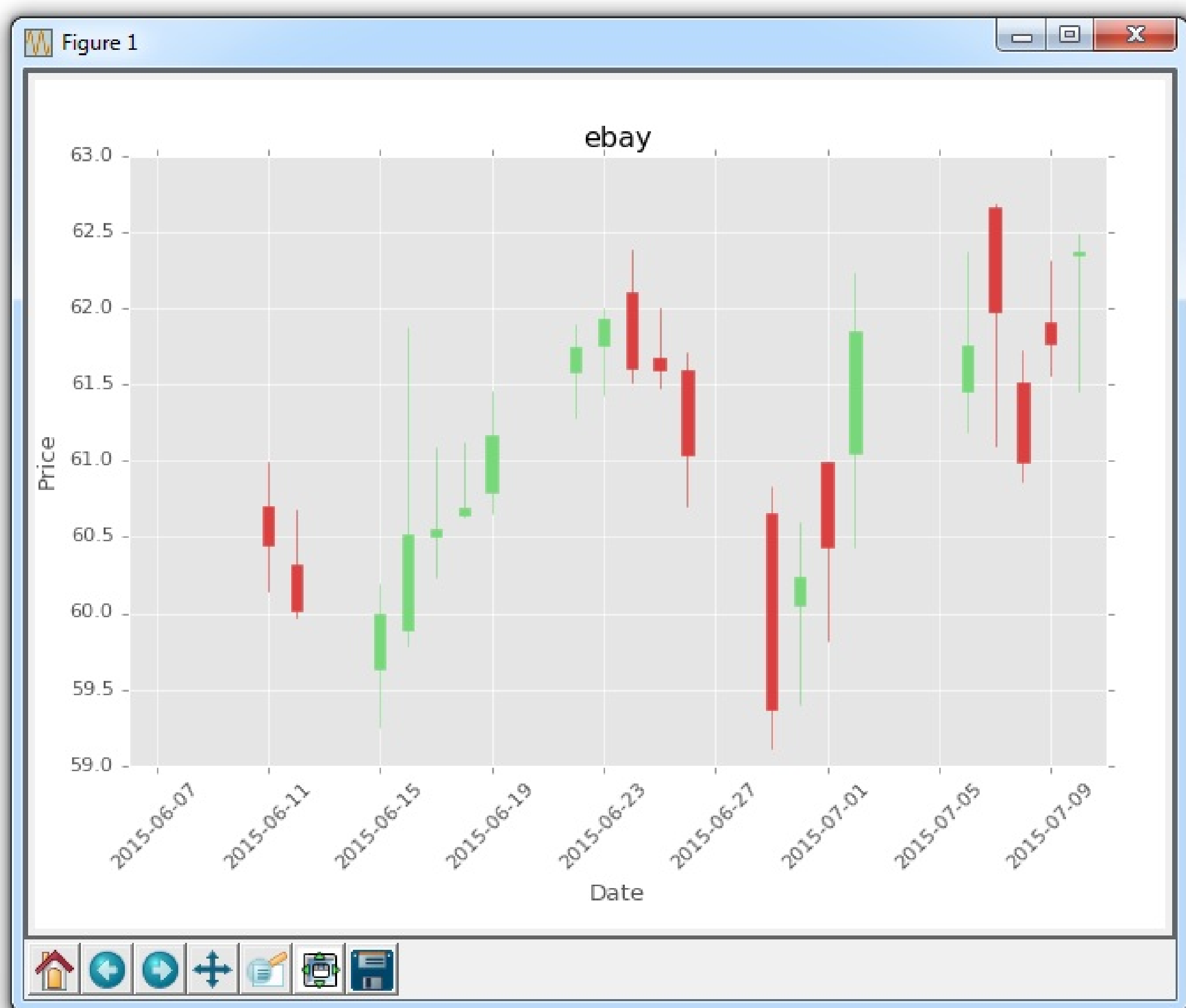
样式页的想法是将自定义样式写入文件，然后，为了使用这些更改并将其应用于图形，所有你需要做的就是导入样式，然后使用该特定样式。 这样，让我们假设你发现自己总是改变图形的各种元素。 你不必为每个图表编写 25 ~ 200 行自定义代码，只需将其写入一个样式，然后加载该样式，并以两行应用所有这些更改即可！让我们开始吧。

```
from matplotlib import style
```

接下来，我们指定要使用的样式。 Matplotlib 已经有了几种样式。

我们可以这样来使用样式：

```
style.use('ggplot')
```

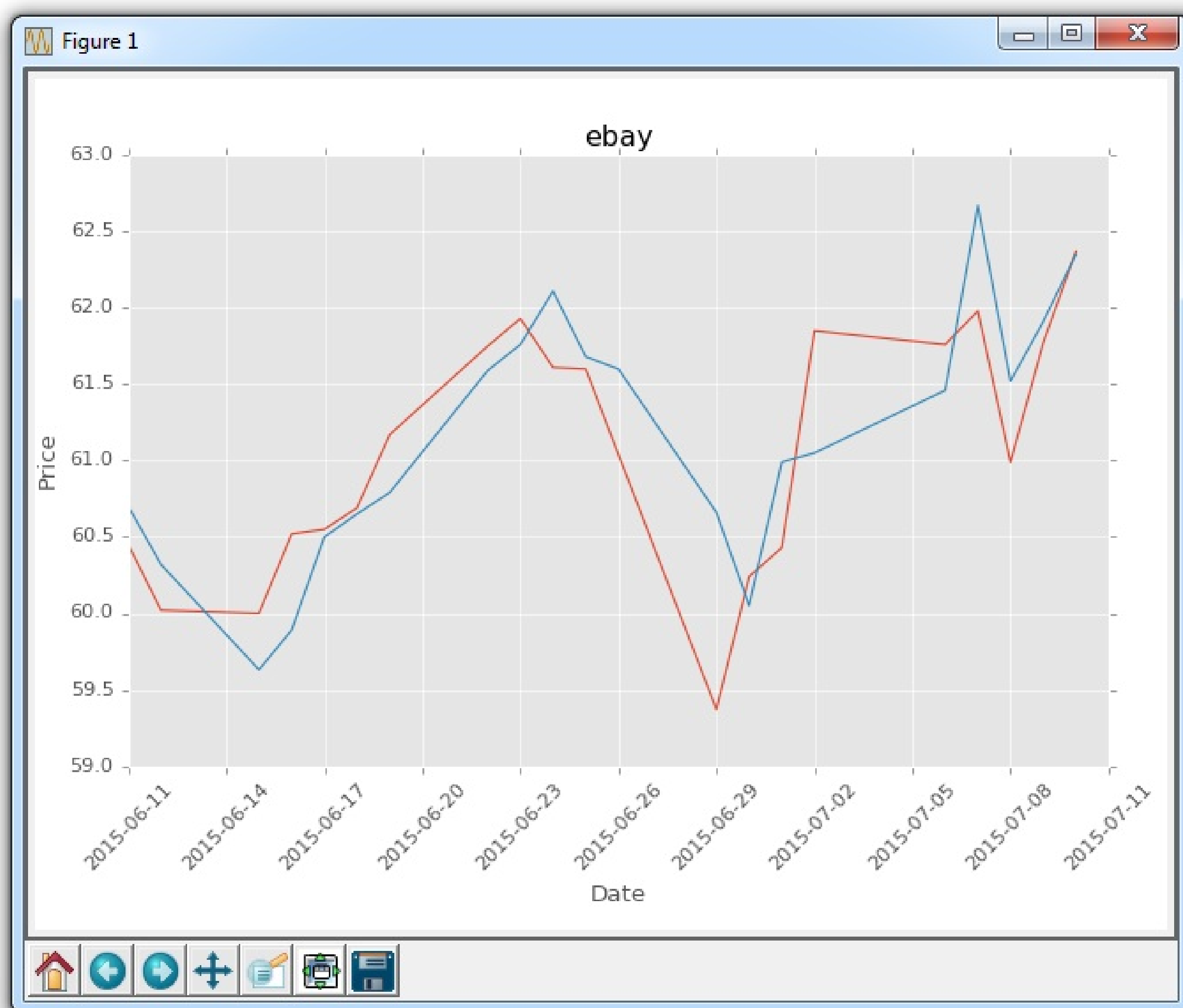


除了标题，标签的颜色是灰色的，轴域的背景是浅灰色，我们可以立即分辨字体是不同的。 我们还注意到，网格实际上是一个白色的实线。 我们的 K 线图保持不变，主要是因为我们在事后定制它。 在样式中加载时，更改会生效，但如果在加载样式后编写新的自定义代码，你的更改也会生效。

因为我们试图展示样式模块，但是让我们继续，简单绘制几行，并暂且注释掉 K 线图：

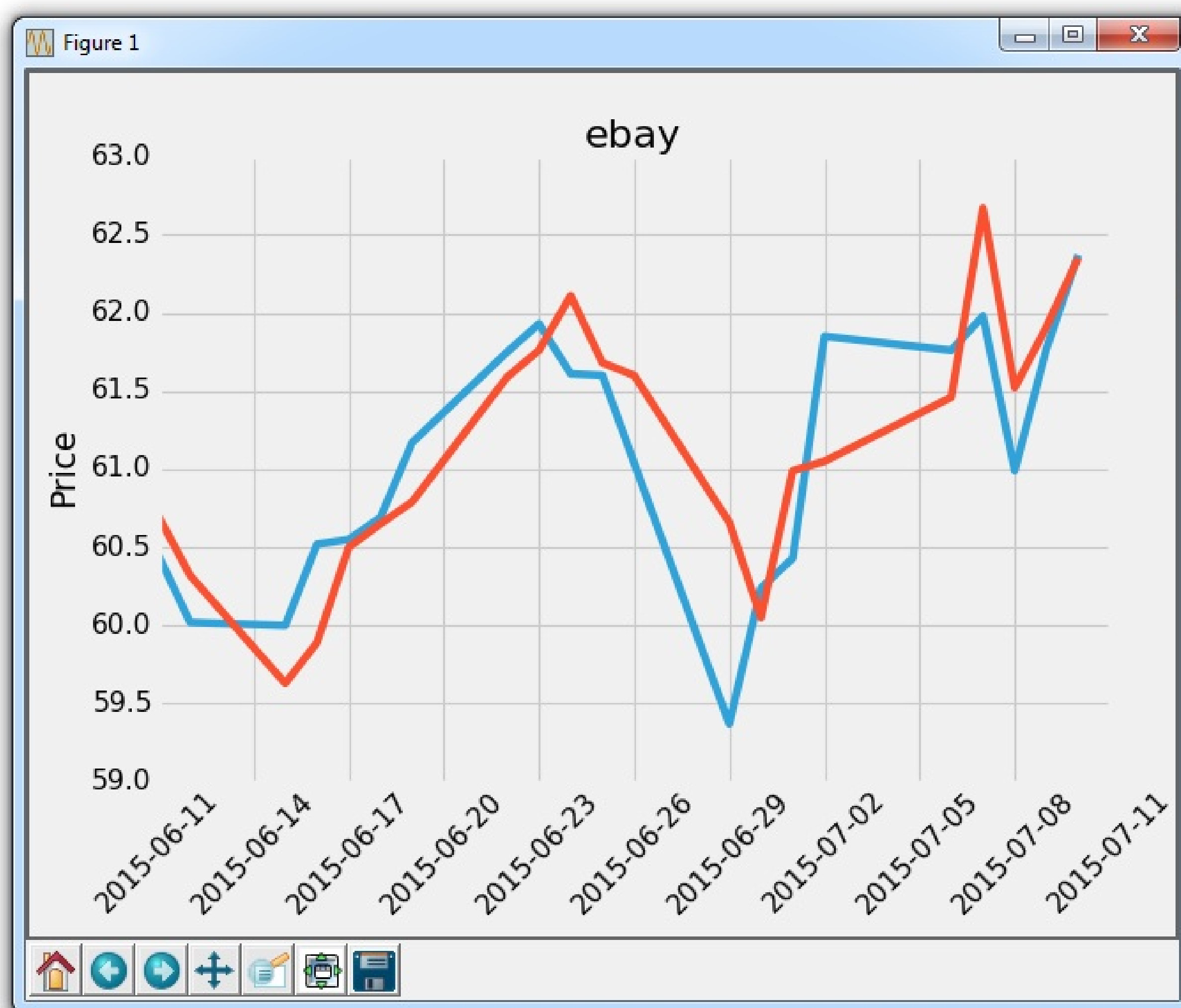
```
#candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', color
down='#db3f3f')
ax1.plot(date,closep)
ax1.plot(date,openp)
```

会生成：



已经比默认值好多了！

样式的另一个例子是 `fivethirtyeight` ：



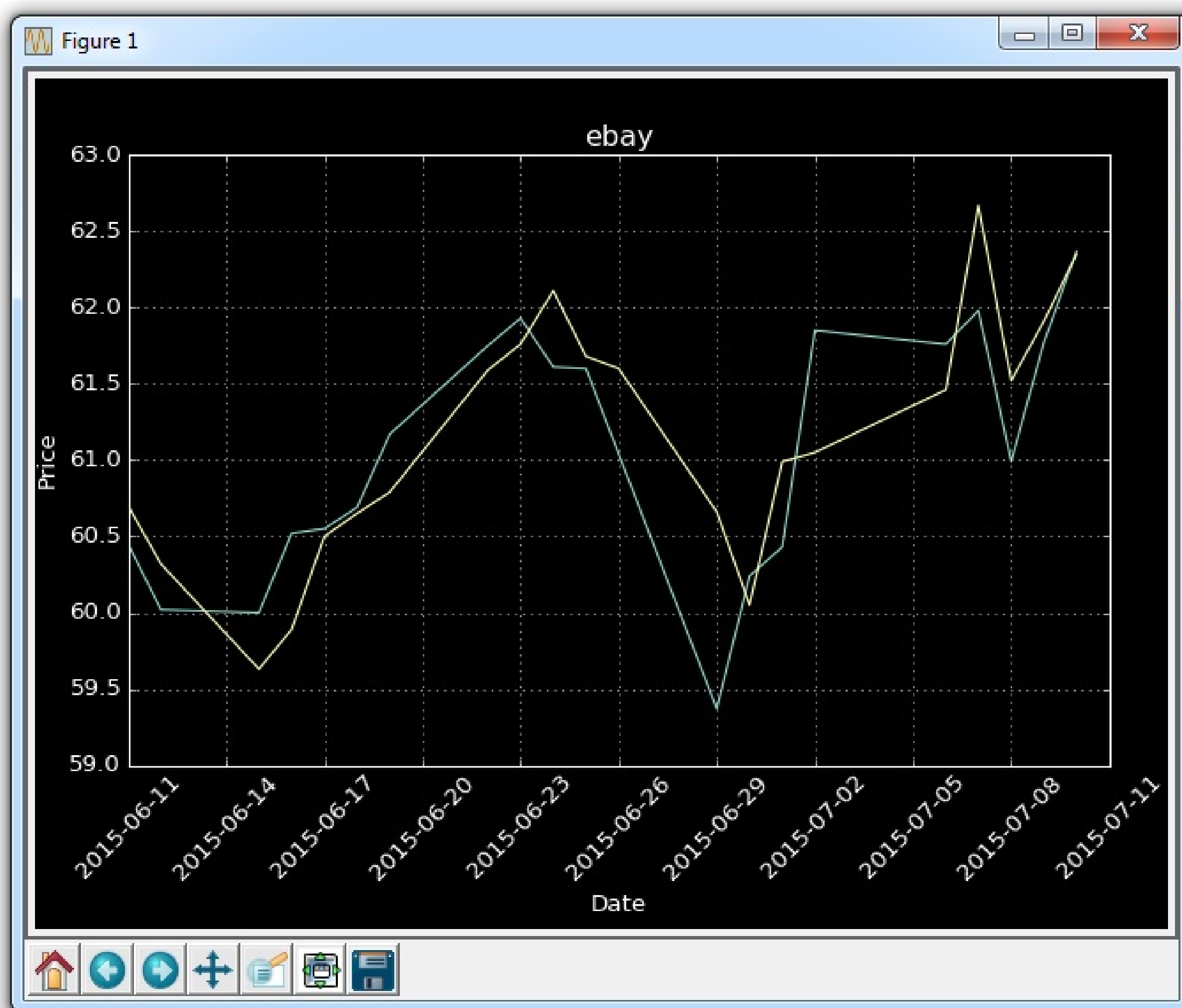
你可以这样查看所有的可用样式：

```
print(plt.style.available)
```

我这里它提供了 `['bmh', 'dark_background', 'ggplot', 'fivethirtyeight', 'grayscale` 。

让我们尝试 `dark_background` ：

```
style.use('dark_background')
```



现在，如果你想制作自己的风格呢？ 首先，你需要找到样式目录。 为了实现它，如果你知道它在哪里，你可以前往你的 matplotlib 目录，或者你可以找到该目录。如果你不知道如何找到该目录，你可以执行以下操作：

```
print(plt.__file__)
```

这至少会告诉你 `pyplot` 模块的位置。

在 matplotlib 目录中，你需要寻找 `mpl-data` 。 然后在那里，你需要寻找 `stylelib` 。 在 Windows 上 ，我的完整路径是： `C:\Python34\Lib\site-packages\matplotlib\mpl-data\stylelib` 。

那里应该显示了所有可用的 `.mplstyle` 文件。 你可以编辑、复制或重命名它们，然后在那里修改为你想要的东西。 然后，无论你用什么来命名 `.mplstyle` 文件，都要放在 `style.use` 中。

第十六章 实时图表

> 原文：Live Graphs with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这篇 Matplotlib 教程中，我们将介绍如何创建实时更新图表，可以在数据源更新时更新其图表。 你可能希望将此用于绘制股票实时定价数据，或者可以将传感器连接到计算机，并且显示传感器实时数据。 为此，我们使用 Matplotlib 的动画功能。

最开始：

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style
```

这里，唯一的新增导入是 `matplotlib.animation as animation` 。 这是一个模块，允许我们在显示之后对图形进行动画处理。

接下来，我们添加一些你熟悉的代码，如果你一直关注这个系列：

```
style.use('fivethirtyeight')

fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
```

现在我们编写动画函数：

```
def animate(i):
    graph_data = open('example.txt','r').read()
    lines = graph_data.split('\n')
    xs = []
    ys = []
    for line in lines:
        if len(line) > 1:
            x, y = line.split(',')
            xs.append(x)
            ys.append(y)
    ax1.clear()
    ax1.plot(xs, ys)
```

我们在这里做的是构建数据，然后绘制它。 注意我们这里不调用 `plt.show()` 。 我们从一个示例文件读取数据，其内容如下：

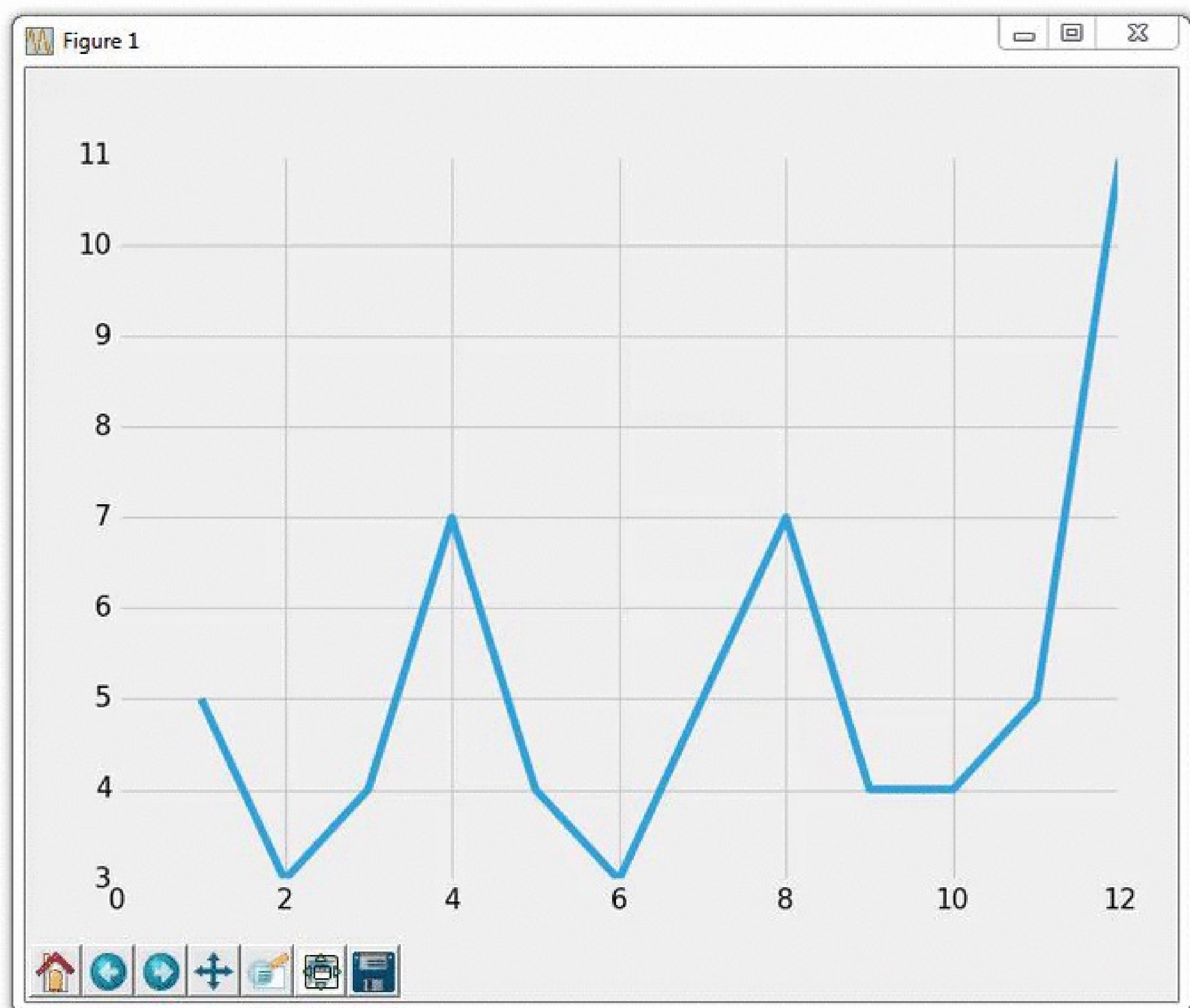
```
1,5
2,3
3,4
4,7
5,4
6,3
7,5
8,7
9,4
10,4
```

我们打开上面的文件，然后存储每一行，用逗号分割成 `xs` 和 `ys`，我们将要绘制它。 然后：

```
ani = animation.FuncAnimation(fig, animate, interval=1000)
plt.show()
```

我们运行动画，将动画放到图表中（`fig`），运行 `animate` 的动画函数，最后我们设置了 1000 的间隔，即 1000 毫秒或 1 秒。

运行此图表的结果应该像往常一样生成图表。 然后，你应该能够使用新的坐标更新 `example.txt` 文件。 这样做会生成一个自动更新的图表，如下：


Figure 1
11
10
9
8
7
6
5
4
3
0
2
4
6
8
10
12

第十七章 注解和文本

> 原文：[Annotations and Text with Matplotlib](https://pythonprogramming.net/annotations-text-matplotlib-tutorial/)
>
> 译者：飞龙
>
>

在本教程中，我们将讨论如何向 Matplotlib 图形添加文本。 我们可以通过两种方式来实现。 一种是将文本放置在图表上的某个位置。 另一个是专门注解图表上的绘图，来引起注意。

这里的起始代码是教程 15，它在这里：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
```

```
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1


    candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax1.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax1.grid(True)


    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('ebay')
```

所以这里是 Yahoo Finance API 的 eBay 的 OHLC K 线图。 这里我们要讲解的第一件事是向图形添加文本。

```
font_dict = {'family':'serif',
             'color':'darkred',
             'size':15}
ax1.text(date[10], closep[1],'Text Example', fontdict=font_dict)
```

在这里，我们需要做一些事情。 首先，我们使用 `ax1.text` 添加文本。 我们使用我们的数据，以坐标形式给出此文本的位置。 首先给出文本的坐标，然后给出要放置的实际文本。 接下来，我们使用 `fontdict` 参数添加一个数据字典，来使用所用的字体。 在我们的字体字典中，我们将字体更改为 `serif` ，颜色为『深红色』，然后将字体大小更改为 15。这将全部应用于我们的图表上的文本，如下所示：



太棒了，接下来我们可以做的是，注解某个特定的绘图。 我们希望这样做来给出更多的信息。 在 eBay 的例子中，也许我们想解释某个具体绘图，或给出一些关于发生了什么的信息。 在股价的例子中，也许有一些发生的新闻会影响价格。 你可以注解新闻来自哪里，这将有助于解释定价变化。

```
ax1.annotate('Bad News!',(date[9],highp[9]),
             xytext=(0.8, 0.9), textcoords='axes fraction',
             arrowprops = dict(facecolor='grey',color='grey'))
```

这里，我们用 `ax1.annotate` 来注解。我们首先传递我们想要注解的文本，然后传递我们让这个注解指向的坐标。我们这样做，是因为当我们注释时，我们可以绘制线条和指向特定点的箭头。接下来，我们指定 `xytext` 的位置。它可以是像我们用于文本放置的坐标位置，但是让我们展示另一个例子。它可以为轴域小数，所以我们使用 0.8 和 0.9。这意味着文本的位置在 `x` 轴的80%和 `y` 轴的90%处。这样，如果我们移动图表，文本将保持在相同位置。

执行它，会生成：



根据你学习这个教程的时间，所指向的点可能有所不同，这只是一个注解的例子，其中有一些合理的想法，即为什么我们需要注解一些东西。

当图表启动时，请尝试单击平移按钮（蓝色十字），然后移动图表。你会看到文本保持不动，但箭头跟随移动并继续指向我们想要的具体的点。这很酷吧！

最后一个图表的完整代码：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
```

```
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1


    candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax1.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax1.grid(True)
    ax1.annotate('Bad News!',(date[9],highp[9]),
                 xytext=(0.8, 0.9), textcoords='axes fraction',
                 arrowprops = dict(facecolor='grey',color='grey'
))

##    # Text placement example:
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    ax1.text(date[10], closep[1],'Text Example', fontdict=font
_dict)

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    #plt.legend()
    plt.subplots_adjust(left=0.09, bottom=0.20, right=0.94, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('ebay')
```

现在，使用注解，我们可以做一些其他事情，如注解股票图表的最后价格。 这就是我们接下来要做的。

第十八章 注解股票图表的最后价格

> 原文：Annotating Last Price Stock Chart with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们将展示如何跟踪股票的最后价格的示例，通过将其注解到轴域的右侧，就像许多图表应用程序会做的那样。

虽然人们喜欢在他们的实时图表中看到历史价格，他们也想看到最新的价格。 大多数应用程序做的是，在价格的 `y` 轴高度处注释最后价格，然后突出显示它，并在价格变化时，在框中将其略微移动。 使用我们最近学习的注解教程，我们可以添加一个 `bbox` 。

我们的核心代码是：

```
bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

ax1.annotate(str(closep[-1]), (date[-1], closep[-1]),
             xytext = (date[-1]+4, closep[-1]), bbox=bbox_props)
```

我们使用 `ax1.annotate` 来放置最后价格的字符串值。 我们不在这里使用它，但我们将要注解的点指定为图上最后一个点。 接下来，我们使用 `xytext` 将我们的文本放置到特定位置。 我们将它的 `y` 坐标指定为最后一个点的 `y` 坐标， `x` 坐标指定为最后一个点的 `x` 坐标，再加上几个点。我们这样做是为了将它移出图表。将文本放在图形外面就足够了，但现在它只是一些浮动文本。

我们使用 `bbox` 参数在文本周围创建一个框。 我们使用 `bbox_props` 创建一个属性字典，包含盒子样式，然后是白色（ `w` ）前景色，黑色（ `k` ）边框颜色并且线宽为 1。 更多框样式请参阅 matplotlib 注解文档 。

最后，这个注解向右移动，需要我们使用 `subplots_adjust` 来创建一些新空间：

```
plt.subplots_adjust(left=0.11, bottom=0.24, right=0.87, top=0.90
, wspace=0.2, hspace=0)
```

这里的完整代码如下：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style
```

```
import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
```

```
        x+=1


    candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax1.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax1.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax1.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+3, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax1.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax1.text(date[10], closep[1],'Text Example', fontdict=font
_dict)

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    #plt.legend()
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.87, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

结果为：


Figure 1
EBAY
63.0
62.5
62.0
61.5
61.0
60.5
60.0
59.5
59.0
Price
62.37
2015-06-07
2015-06-11
2015-06-15
2015-06-19
2015-06-23
2015-06-27
2015-07-01
2015-07-05
2015-07-09
Date

第十九章 子图

> 原文：Subplots with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们将讨论子图。 有两种处理子图的主要方法，用于在同一图上创建多个图表。 现在，我们将从一个干净的代码开始。 如果你一直关注这个教程，那么请确保保留旧的代码，或者你可以随时重新查看上一个教程的代码。

首先，让我们使用样式，创建我们的图表，然后创建一个随机创建示例绘图的函数：

```
import random
import matplotlib.pyplot as plt
from matplotlib import style

style.use('fivethirtyeight')

fig = plt.figure()

def create_plots():
    xs = []
    ys = []

    for i in range(10):
        x = i
        y = random.randrange(10)

        xs.append(x)
        ys.append(y)
    return xs, ys
```

现在，我们开始使用 `add_subplot` 方法创建子图：

```
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(212)
```

它的工作原理是使用 3 个数字，即：行数（ `numRows` ）、列数（ `numCols` ）和绘图编号（ `plotNum` ）。

所以，221 表示两行两列的第一个位置。222 是两行两列的第二个位置。最后，212 是两行一列的第二个位置。

```
2x2：

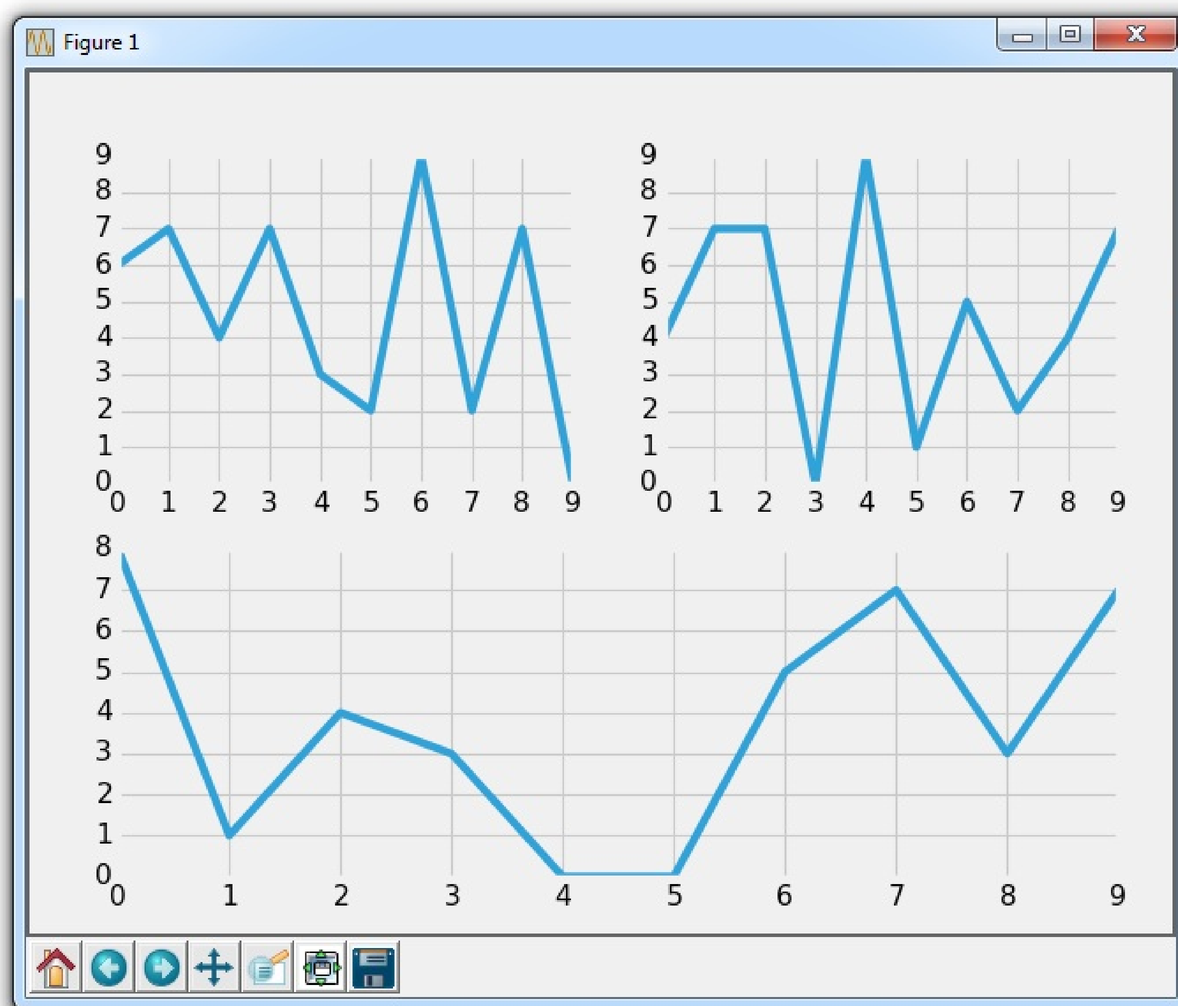
+-----+-----+
|  1  |  2  |
+-----+-----+
|  3  |  4  |
+-----+-----+

2x1：

+-----------+
|     1     |
+-----------+
|     2     |
+-----------+
```

> 译者注：原文此处表述有误，译文已更改。
>
> 译者注： `221` 是缩写形式，仅在行数乘列数小于 10 时有效，否则要写成 `2,2,1` 。

此代码结果为：



这就是 `add_subplot` 。尝试一些你认为可能很有趣的配置，然后尝试使用 `add_subplot` 创建它们，直到你感到满意。

接下来，让我们介绍另一种方法，它是 `subplot2grid` 。

删除或注释掉其他轴域定义，然后添加：

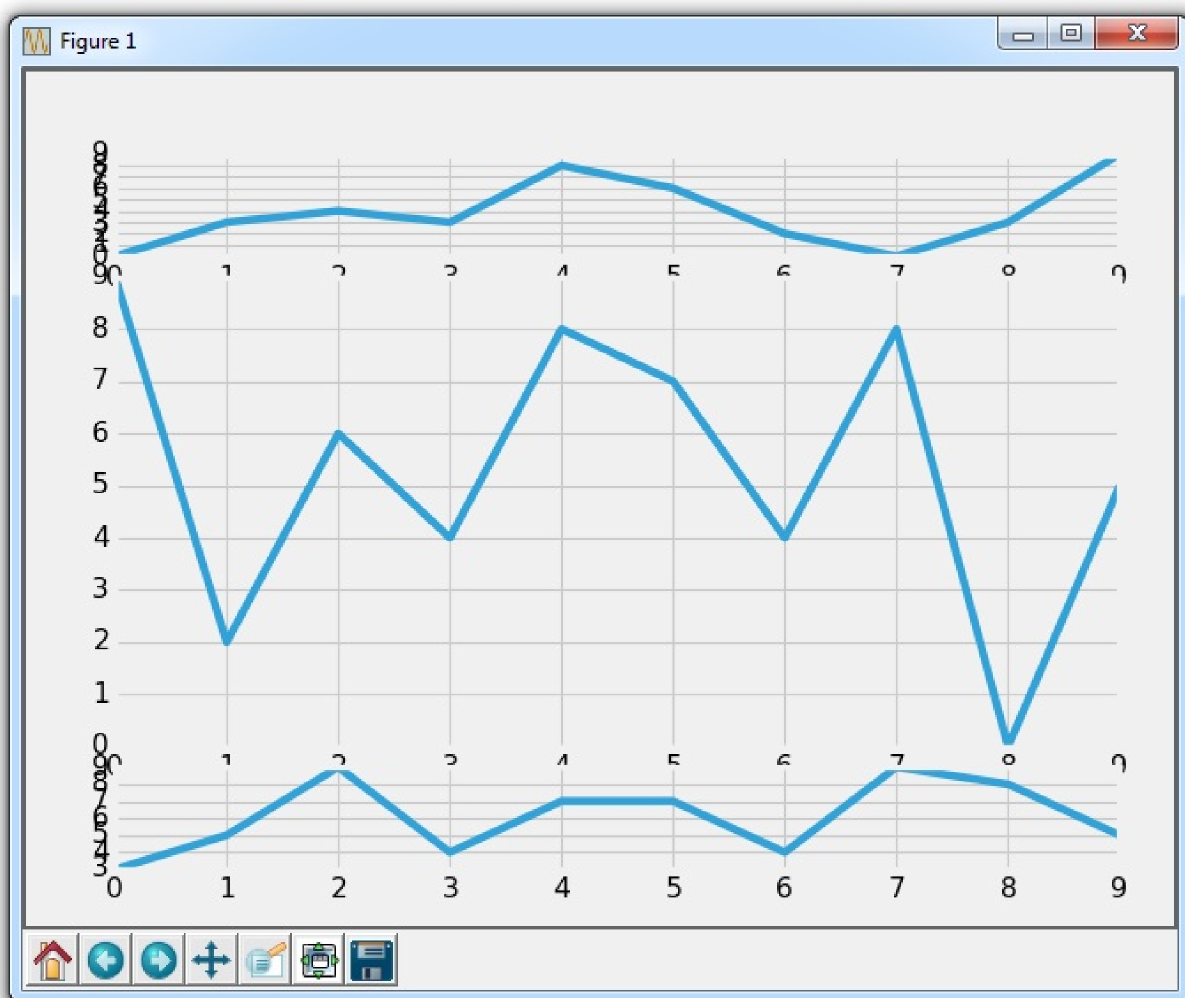
```
ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)
```

所以， `add_subplot` 不能让我们使一个绘图覆盖多个位置。但是这个新的 `subplot2grid` 可以。所以， `subplot2grid` 的工作方式是首先传递一个元组，它是网格形状。我们传递了 `(6,1)` ，这意味着整个图表分为六行一列。下一个元组是左上角的起始点。对于 `ax1` ，这是 `0,0` ，因此它起始于顶部。接下来，我们可以选择指定 `rowspan` 和 `colspan` 。这是轴域所占的行数和列数。

```
6x1：

             colspan=1
(0,0)   +-----------+
        |    ax1    | rowspan=1
(1,0)   +-----------+
        |           |
        |    ax2    | rowspan=4
        |           |
        |           |
(5,0)   +-----------+
        |    ax3    | rowspan=1
        +-----------+
```

结果为：



显然，我们在这里有一些重叠的问题，我们可以调整子图来处理它。

再次，尝试构思各种配置的子图，使用 `subplot2grid` 制作出来，直到你感到满意！

我们将继续使用 `subplot2grid` ，将它应用到我们已经逐步建立的代码中，我们将在下一个教程中继续。

第二十章 将子图应用于我们的图表

> 原文：Implementing Subplots to our Chart with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们将处理我们以前教程的代码，并实现上一个教程中的子图配置。 我们的起始代码是这样：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((1,1), (0,0))

    stock_price_url = 'http://chartapi.finance.yahoo.com/instrument/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
    source_code = urllib.request.urlopen(stock_price_url).read().decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
```

```
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1


    candlestick_ohlc(ax1, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax1.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax1.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax1.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax1.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax1.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
```

```
##    # Hard coded text
##    ax1.text(date[10], closep[1],'Text Example', fontdict=font
_dict)

    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.title(stock)
    #plt.legend()
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

一个主要的改动是修改轴域的定义：

```
ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
plt.title(stock)
ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
plt.xlabel('Date')
plt.ylabel('Price')
ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)
```

现在，`ax2` 是我们实际上在绘制的股票价格数据。顶部和底部图表将作为指标信息。

在我们绘制数据的代码中，我们需要将 `ax1` 更改为 `ax2`：

```
candlestick_ohlc(ax2, ohlc, width=0.4, colorup='#77d879', colord
own='#db3f3f')

for label in ax2.xaxis.get_ticklabels():
    label.set_rotation(45)

ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
ax2.xaxis.set_major_locator(mticker.MaxNLocator(10))
ax2.grid(True)

bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
             xytext = (date[-1]+4, closep[-1]), bbox=bbox_props)
```

更改之后，代码为：

```
import matplotlib.pyplot as plt
```

```
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
    plt.xlabel('Date')
    plt.ylabel('Price')
    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1m/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
```

```
                                                                    conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1


    candlestick_ohlc(ax2, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax2.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax2.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax1.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax1.text(date[10], closep[1],'Text Example', fontdict=font
_dict)


    #
    #plt.legend()
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
```

```
    plt.show()


graph_data('EBAY')
```

结果为：



第二十一章 更多指标数据

> 原文：More indicator data with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这篇 Matplotlib 教程中，我们介绍了添加一些简单的函数来计算数据，以便我们填充我们的轴域。 一个是简单的移动均值，另一个是简单的价格 HML 计算。

这些新函数是：

```
def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas

def high_minus_low(highs, lows):
    return highs-lows
```

你不需要太过专注于理解移动均值的工作原理，我们只是对样本数据来计算它，以便可以学习更多自定义 Matplotlib 的东西。

我们还想在脚本顶部为移动均值定义一些值：

```
MA1 = 10
MA2 = 30
```

下面，在我们的 `graph_data` 函数中：

```
ma1 = moving_average(closep,MA1)
ma2 = moving_average(closep,MA2)
start = len(date[MA2-1:])

h_l = list(map(high_minus_low, highp, lowp))
```

在这里，我们计算两个移动均值和 HML。

我们还定义了一个『起始』点。 我们这样做是因为我们希望我们的数据排成一行。 例如，20 天的移动均值需要 20 个数据点。 这意味着我们不能在第 5 天真正计算 20 天的移动均值。 因此，当我们计算移动均值时，我们会失去一些数据。 为了处理这种数据的减法，我们使用起始变量来计算应该有多少数据。 这里，我们可以安全地使用 `[-start:]` 绘制移动均值，并且如果我们希望的话，对所有绘图进行上述步骤来排列数据。

接下来，我们可以在 `ax1` 上绘制 HML，通过这样：

```
ax1.plot_date(date,h_l,'-')
```

最后我们可以通过这样向 `ax3` 添加移动均值：

```
ax3.plot(date[-start:], ma1[-start:])
ax3.plot(date[-start:], ma2[-start:])
```

我们的完整代码，包括增加我们所用的时间范围：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30

def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
```

```
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
    plt.xlabel('Date')
    plt.ylabel('Price')
    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))

    ax1.plot_date(date,h_l,'-')


    candlestick_ohlc(ax2, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')

    for label in ax2.xaxis.get_ticklabels():
```

```
        label.set_rotation(45)

    ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax2.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
_dict)

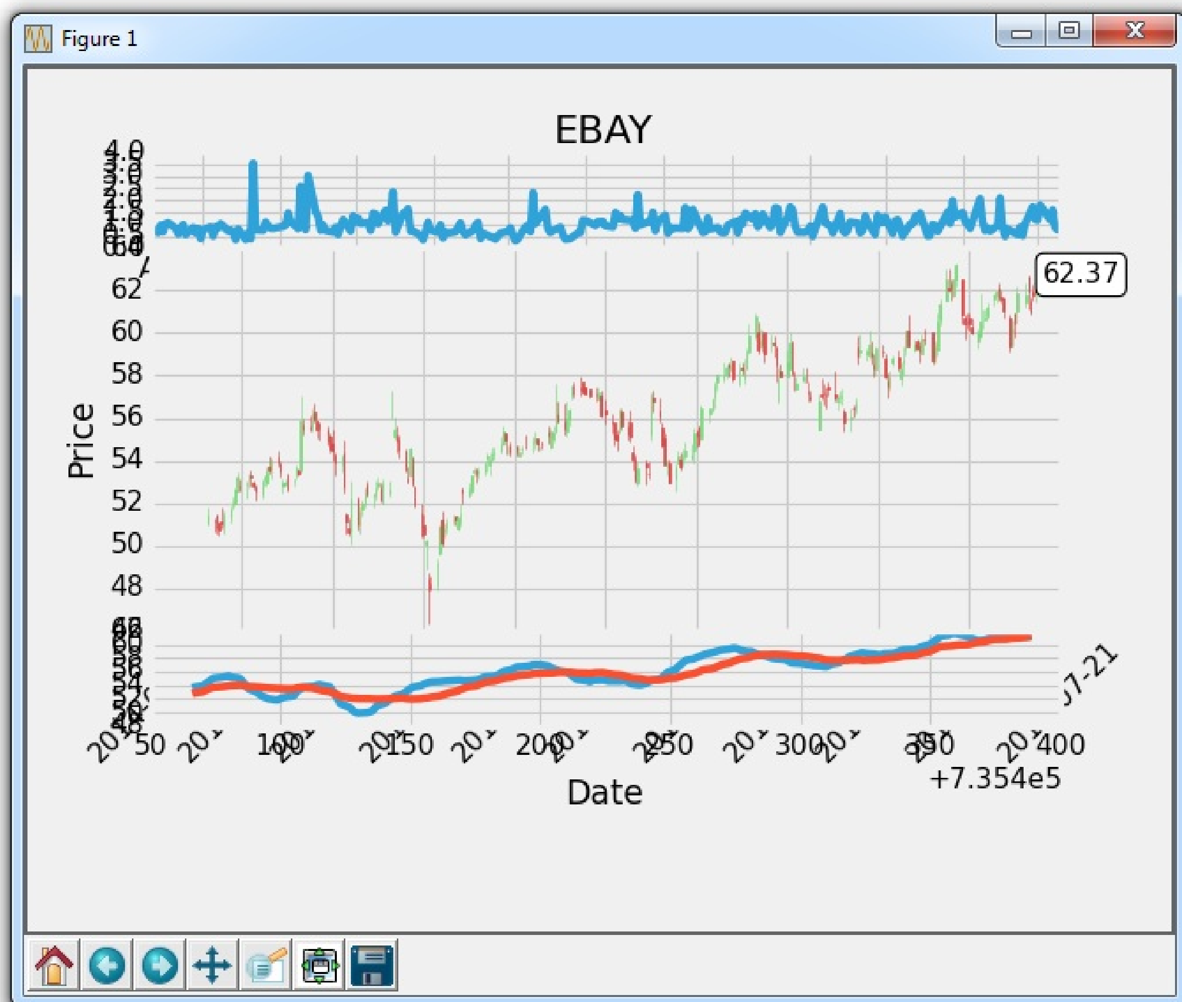


    ax3.plot(date[-start:], ma1[-start:])
    ax3.plot(date[-start:], ma2[-start:])


    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

代码效果如图：


Figure 1
EBAY
Price
62.37
Date
+7.354e5

第二十二章 自定义填充、修剪和清除

> 原文：Custom fills, pruning, and cleaning with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

欢迎阅读另一个 Matplotlib 教程！ 在本教程中，我们将清除图表，然后再做一些自定义。

我们当前的代码是：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30

def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):
```

```
    fig = plt.figure()
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
    plt.xlabel('Date')
    plt.ylabel('Price')
    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                           delimi
ter=',',
                                                           unpack=
True,
                                                           conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))

    ax1.plot_date(date,h_l,'-')


    candlestick_ohlc(ax2, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')
```

```
    for label in ax2.xaxis.get_ticklabels():
        label.set_rotation(45)

    ax2.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax2.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
_dict)



    ax3.plot(date[-start:], ma1[-start:])
    ax3.plot(date[-start:], ma2[-start:])


    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

现在我认为向我们的移动均值添加自定义填充是一个很好的主意。 移动均值通常用于说明价格趋势。 这个想法是，你可以计算一个快速和一个慢速的移动均值。 一般来说，移动均值用于使价格变得『平滑』。 他们总是『滞后』于价格，但是我们的想法是计算不同的速度。 移动均值越大就越『慢』。 所以这个想法是，如果『较快』的移动均值超过『较慢』的均值，那么价格就会上升，这是一件好事。 如

果较快的 MA 从较慢的 MA 下方穿过，则这是下降趋势并且通常被视为坏事。 我的想法是在快速和慢速 MA 之间填充，『上升』趋势为绿色，然后下降趋势为红色。方法如下：

```
ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                 where=(ma1[-start:] < ma2[-start:]),
                 facecolor='r', edgecolor='r', alpha=0.5)

ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                 where=(ma1[-start:] > ma2[-start:]),
                 facecolor='g', edgecolor='g', alpha=0.5)
```

下面，我们会碰到一些我们可解决的问题：

```
ax3.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
ax3.xaxis.set_major_locator(mticker.MaxNLocator(10))

for label in ax3.xaxis.get_ticklabels():
    label.set_rotation(45)

plt.setp(ax1.get_xticklabels(), visible=False)
plt.setp(ax2.get_xticklabels(), visible=False)
```

这里，我们剪切和粘贴 `ax2` 日期格式，然后我们将 `x` 刻度标签设置为 `false`，去掉它们！

我们还可以通过在轴域定义中执行以下操作，为每个轴域提供自定义标签：

```
fig = plt.figure()
ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
plt.title(stock)
ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
plt.xlabel('Date')
plt.ylabel('Price')
ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)
```

接下来，我们可以看到，我们 `y` 刻度有许多数字，经常互相覆盖。 我们也看到轴之间互相重叠。 我们可以这样：

```
ax1.yaxis.set_major_locator(mticker.MaxNLocator(nbins=5, prune='
lower'))
```

所以，这里发生的是，我们通过首先将 `nbins` 设置为 5 来修改我们的 `y` 轴对象。这意味着我们显示的标签最多为 5 个。然后我们还可以『修剪』标签，因此，在我们这里， 我们修剪底部标签，这会使它消失，所以现在不会有任何文本重叠。

我们仍然可能打算修剪 `ax2` 的顶部标签，但这里是我们目前为止的源代码：

当前的源码：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30

def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    plt.ylabel('H-L')
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1)
    plt.ylabel('Price')
    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1)
    plt.ylabel('MAvgs')


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
```

```
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))


    ax1.plot_date(date,h_l,'-')
    ax1.yaxis.set_major_locator(mticker.MaxNLocator(nbins=5, pru
ne='lower'))


    candlestick_ohlc(ax2, ohlc, width=0.4, colorup='#77d879', co
lordown='#db3f3f')




    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)
```

```
    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                     xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
_dict)



    ax3.plot(date[-start:], ma1[-start:], linewidth=1)
    ax3.plot(date[-start:], ma2[-start:], linewidth=1)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] < ma2[-start:]),
                     facecolor='r', edgecolor='r', alpha=0.5)

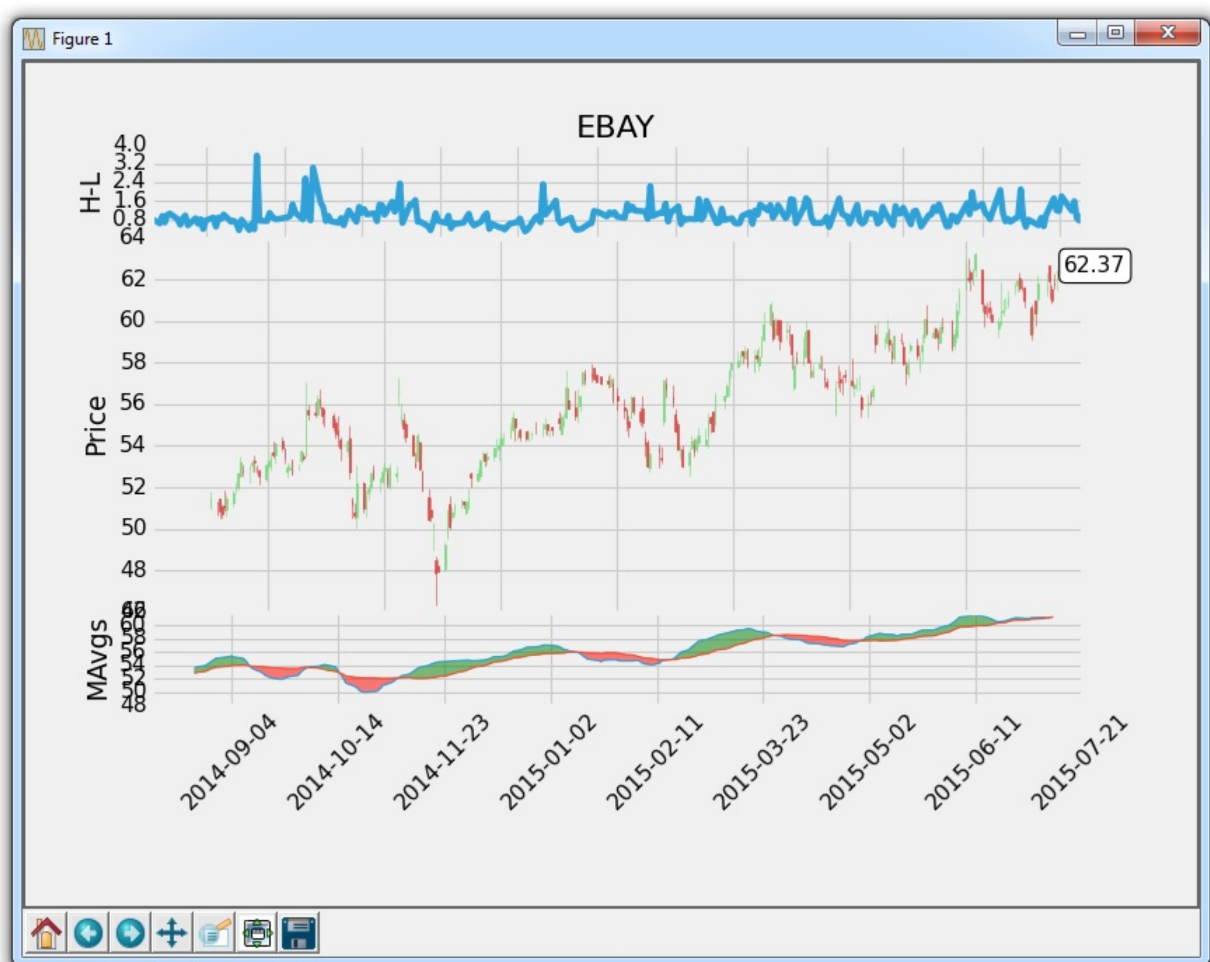
    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] > ma2[-start:]),
                     facecolor='g', edgecolor='g', alpha=0.5)

    ax3.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax3.xaxis.set_major_locator(mticker.MaxNLocator(10))

    for label in ax3.xaxis.get_ticklabels():
        label.set_rotation(45)

    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.setp(ax2.get_xticklabels(), visible=False)
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```



看起来好了一些，但是仍然有一些东西需要清除。

第二十三章 共享 X 轴

> 原文：Share X Axis, sharex, with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 数据可视化教程中，我们将讨论 `sharex` 选项，它允许我们在图表之间共享 `x` 轴。将 `sharex` 看做『复制 x』也许更好。

在我们开始之前，首先我们要做些修剪并在另一个轴上设置最大刻度数，如下所示：

```
ax2.yaxis.set_major_locator(mticker.MaxNLocator(nbins=7, prune='upper'))
```

以及

```
ax3.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, prune='upper'))
```

现在，让我们共享所有轴域之间的 `x` 轴。 为此，我们需要将其添加到轴域定义中：

```
fig = plt.figure()
ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
plt.title(stock)
plt.ylabel('H-L')
ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1, sharex=ax1)
plt.ylabel('Price')
ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1, sharex=ax1)
plt.ylabel('MAvgs')
```

上面，对于 `ax2` 和 `ax3`，我们添加一个新的参数，称为 `sharex`，然后我们说，我们要与 `ax1` 共享 `x` 轴。

使用这种方式，我们可以加载图表，然后我们可以放大到一个特定的点，结果将是这样：



所以这意味着所有轴域沿着它们的 `x` 轴一起移动。 这很酷吧！

接下来，让我们将 `[-start:]` 应用到所有数据，所以所有轴域都起始于相同地方。 我们最终的代码为：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30
```

```
def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    plt.ylabel('H-L')
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1, s
harex=ax1)
    plt.ylabel('Price')
    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1, s
harex=ax1)
    plt.ylabel('MAvgs')


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})
```

```
    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))


    ax1.plot_date(date[-start:],h_l[-start:],'-')
    ax1.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='lower'))


    candlestick_ohlc(ax2, ohlc[-start:], width=0.4, colorup='#77
d879', colordown='#db3f3f')



    ax2.yaxis.set_major_locator(mticker.MaxNLocator(nbins=7, pru
ne='upper'))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)


##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
```

```
_dict)



    ax3.plot(date[-start:], ma1[-start:], linewidth=1)
    ax3.plot(date[-start:], ma2[-start:], linewidth=1)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] < ma2[-start:]),
                     facecolor='r', edgecolor='r', alpha=0.5)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] > ma2[-start:]),
                     facecolor='g', edgecolor='g', alpha=0.5)

    ax3.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax3.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax3.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='upper'))

    for label in ax3.xaxis.get_ticklabels():
        label.set_rotation(45)



    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.setp(ax2.get_xticklabels(), visible=False)
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('EBAY')
```

下面我们会讨论如何创建多个 `y` 轴。

第二十四章 多个 Y 轴

> 原文：Multi Y Axis with twinx Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这篇 Matplotlib 教程中，我们将介绍如何在同一子图上使用多个 Y 轴。 在我们的例子中，我们有兴趣在同一个图表及同一个子图上绘制股票价格和交易量。

为此，首先我们需要定义一个新的轴域，但是这个轴域是 `ax2` 仅带有 `x` 轴的『双生子』。

这足以创建轴域了。我们叫它 `ax2v` ，因为这个轴域是 `ax2` 加交易量。

现在，我们在轴域上定义绘图，我们将添加：

```
ax2v.fill_between(date[-start:],0, volume[-start:], facecolor='#
0079a3', alpha=0.4)
```

我们在 0 和当前交易量之间填充，给予它蓝色的前景色，然后给予它一个透明度。 我们想要应用幽冥毒，以防交易量最终覆盖其它东西，所以我们仍然可以看到这两个元素。

所以，到现在为止，我们的代码为：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30

def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
```

```
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    plt.ylabel('H-L')
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1, s
harex=ax1)
    plt.ylabel('Price')
    ax2v = ax2.twinx()

    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1, s
harex=ax1)
    plt.ylabel('MAvgs')


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
```

```
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))


    ax1.plot_date(date[-start:],h_l[-start:],'-')
    ax1.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='lower'))


    candlestick_ohlc(ax2, ohlc[-start:], width=0.4, colorup='#77
d879', colordown='#db3f3f')



    ax2.yaxis.set_major_locator(mticker.MaxNLocator(nbins=7, pru
ne='upper'))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)

##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
_dict)
    ax2v.fill_between(date[-start:],0, volume[-start:], facecolo
r='#0079a3', alpha=0.4)
```

```
    ax3.plot(date[-start:], ma1[-start:], linewidth=1)
    ax3.plot(date[-start:], ma2[-start:], linewidth=1)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] < ma2[-start:]),
                     facecolor='r', edgecolor='r', alpha=0.5)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] > ma2[-start:]),
                     facecolor='g', edgecolor='g', alpha=0.5)

    ax3.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax3.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax3.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='upper'))

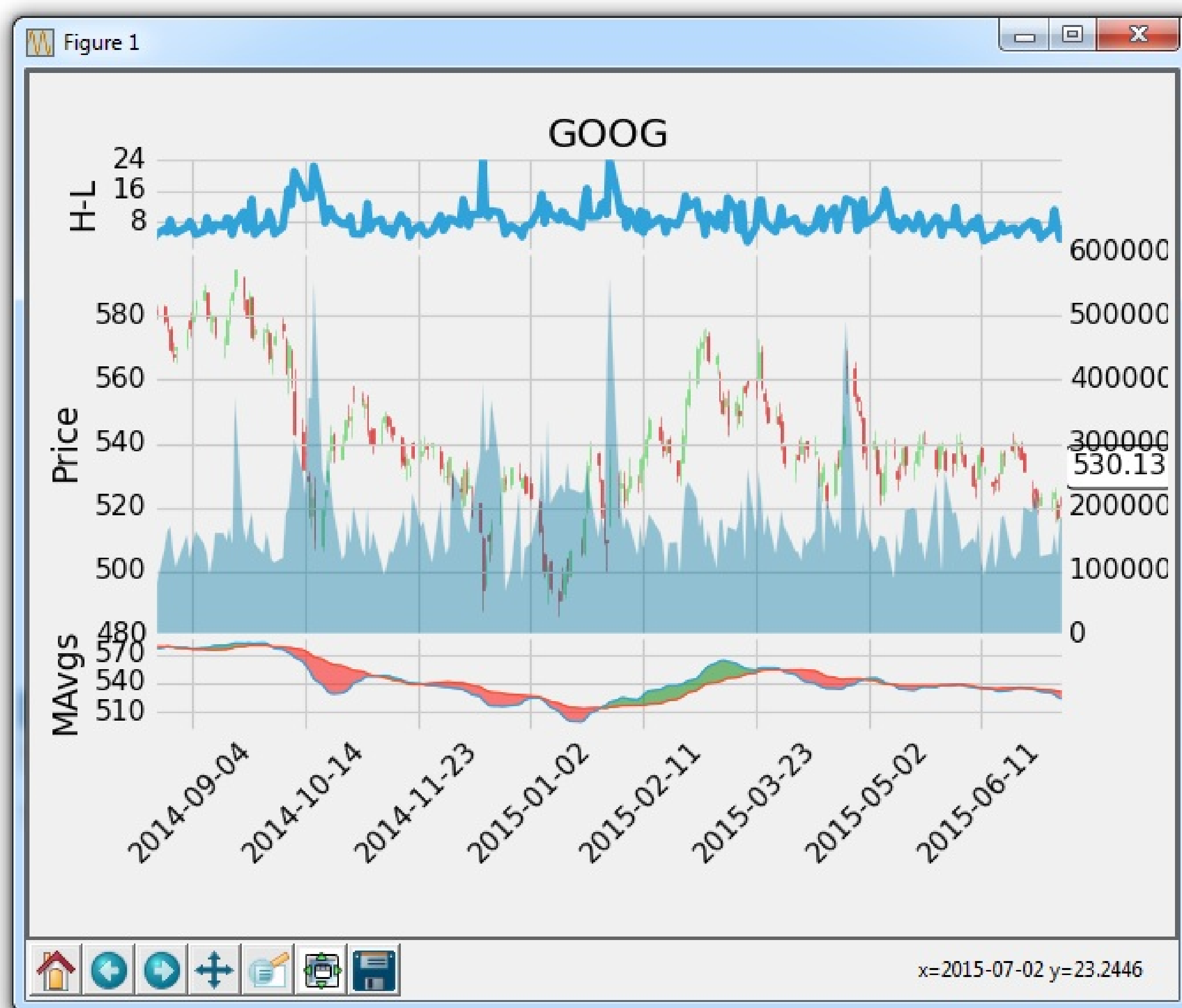
    for label in ax3.xaxis.get_ticklabels():
        label.set_rotation(45)



    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.setp(ax2.get_xticklabels(), visible=False)
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('GOOG')
```

会生成：



太棒了，到目前为止还不错。 接下来，我们可能要删除新 `y` 轴上的标签，然后我们也可能不想让交易量占用太多空间。 没问题：

首先：

```
ax2v.axes.yaxis.set_ticklabels([])
```

上面将 `y` 刻度标签设置为一个空列表，所以不会有任何标签了。

> 译者注：所以将标签删除之后，添加新轴的意义是什么？直接在原轴域上绘图就可以了。

接下来，我们可能要将网格设置为 `false` ，使轴域上不会有双网格：

```
ax2v.grid(False)
```

最后，为了处理交易量占用很多空间，我们可以做以下操作：

```
ax2v.set_ylim(0, 3*volume.max())
```

所以这设置 `y` 轴显示范围从 0 到交易量的最大值的 3 倍。 这意味着，在最高点，交易量最多可占据图形的33%。 所以，增加 `volume.max` 的倍数越多，空间就越小/越少。

现在，我们的图表为：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30

def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure()
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    plt.ylabel('H-L')
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1, s
```

```
harex=ax1)
    plt.ylabel('Price')
    ax2v = ax2.twinx()

    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1, s
harex=ax1)
    plt.ylabel('MAvgs')


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))


    ax1.plot_date(date[-start:],h_l[-start:],'-')
    ax1.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='lower'))
```

```
    candlestick_ohlc(ax2, ohlc[-start:], width=0.4, colorup='#77
d879', colordown='#db3f3f')



    ax2.yaxis.set_major_locator(mticker.MaxNLocator(nbins=7, pru
ne='upper'))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+5, closep[-1]), bbox=bbox_pr
ops)

##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
_dict)

    ax2v.fill_between(date[-start:],0, volume[-start:], facecolo
r='#0079a3', alpha=0.4)
    ax2v.axes.yaxis.set_ticklabels([])
    ax2v.grid(False)
    ax2v.set_ylim(0, 3*volume.max())



    ax3.plot(date[-start:], ma1[-start:], linewidth=1)
    ax3.plot(date[-start:], ma2[-start:], linewidth=1)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] < ma2[-start:]),
                     facecolor='r', edgecolor='r', alpha=0.5)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] > ma2[-start:]),
                     facecolor='g', edgecolor='g', alpha=0.5)

    ax3.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax3.xaxis.set_major_locator(mticker.MaxNLocator(10))
```

```
    ax3.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='upper'))

    for label in ax3.xaxis.get_ticklabels():
        label.set_rotation(45)



    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.setp(ax2.get_xticklabels(), visible=False)
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)
    plt.show()


graph_data('GOOG')
```

到这里，我们差不多完成了。 这里唯一的缺陷是一个好的图例。 一些线条是显而易见的，但人们可能会好奇移动均值的参数是什么，我们这里是 10 和 30。 添加自定义图例是下一个教程中涉及的内容。

第二十五章 自定义图例

> 原文：Custom Legends with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这篇 Matplotlib 教程中，我们将讨论自定义图例。 我们已经介绍了添加图例的基础知识。

图例的主要问题通常是图例阻碍了数据的展示。 这里有几个选项。 一个选项是将图例放在轴域外，但是我们在这里有多个子图，这是非常困难的。 相反，我们将使图例稍微小一点，然后应用一个透明度。

首先，为了创建一个图例，我们需要向我们的数据添加我们想要显示在图例上的标签。

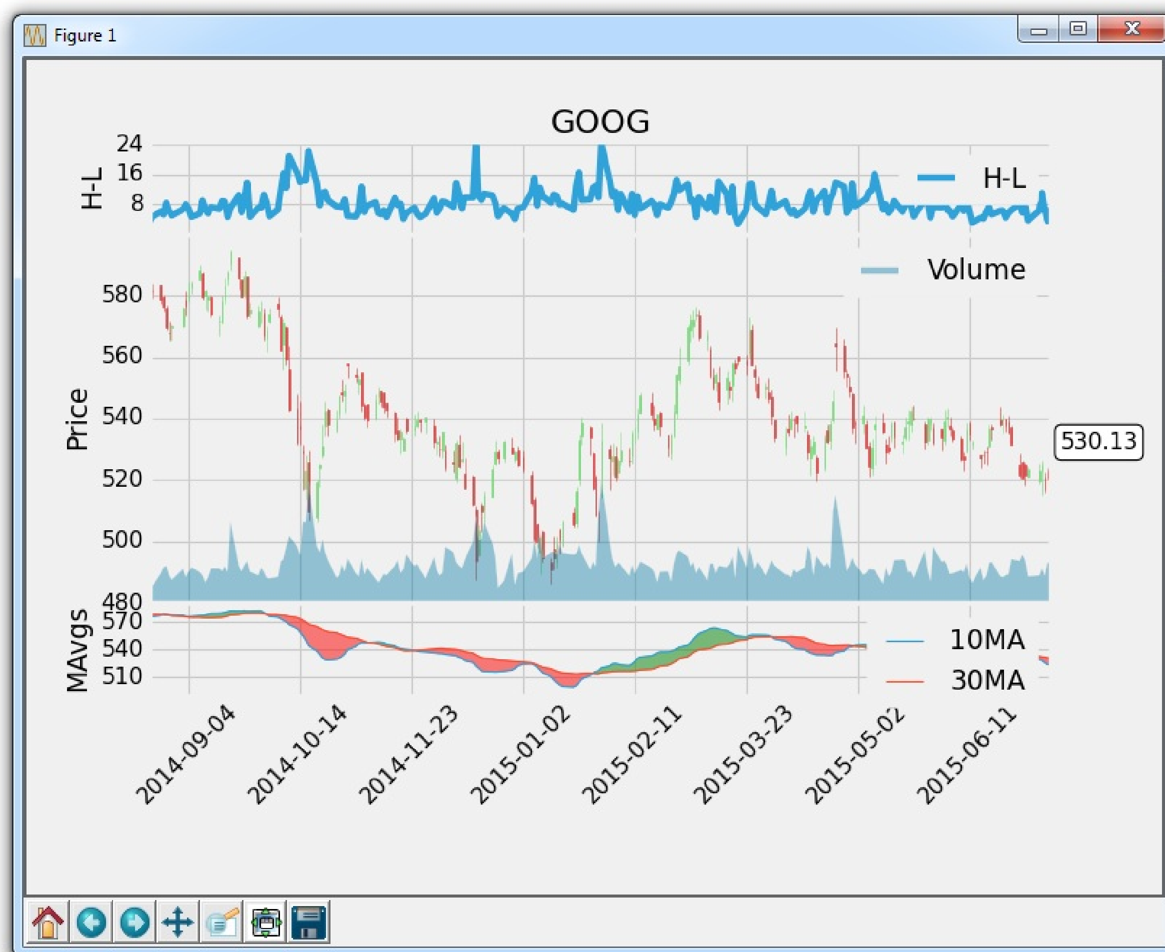
```
ax1.plot_date(date[-start:],h_l[-start:],'-', label='H-L')
...
ax2v.plot([],[], color='#0079a3', alpha=0.4, label='Volume')
...
ax3.plot(date[-start:], ma1[-start:], linewidth=1, label=(str(MA
1)+'MA'))
ax3.plot(date[-start:], ma2[-start:], linewidth=1, label=(str(MA
2)+'MA'))
```

请注意，我们通过创建空行为交易量添加了标签。 请记住，我们不能对任何填充应用标签，所以这就是我们添加这个空行的原因。

现在，我们可以在右下角添加图例，通过在 `plt.show()` 之前执行以下操作：

```
ax1.legend()
ax2v.legend()
ax3.legend()
```

会生成：



所以，我们可以看到，图例还是占用了一些位置。 让我们更改位置，大小并添加透明度：

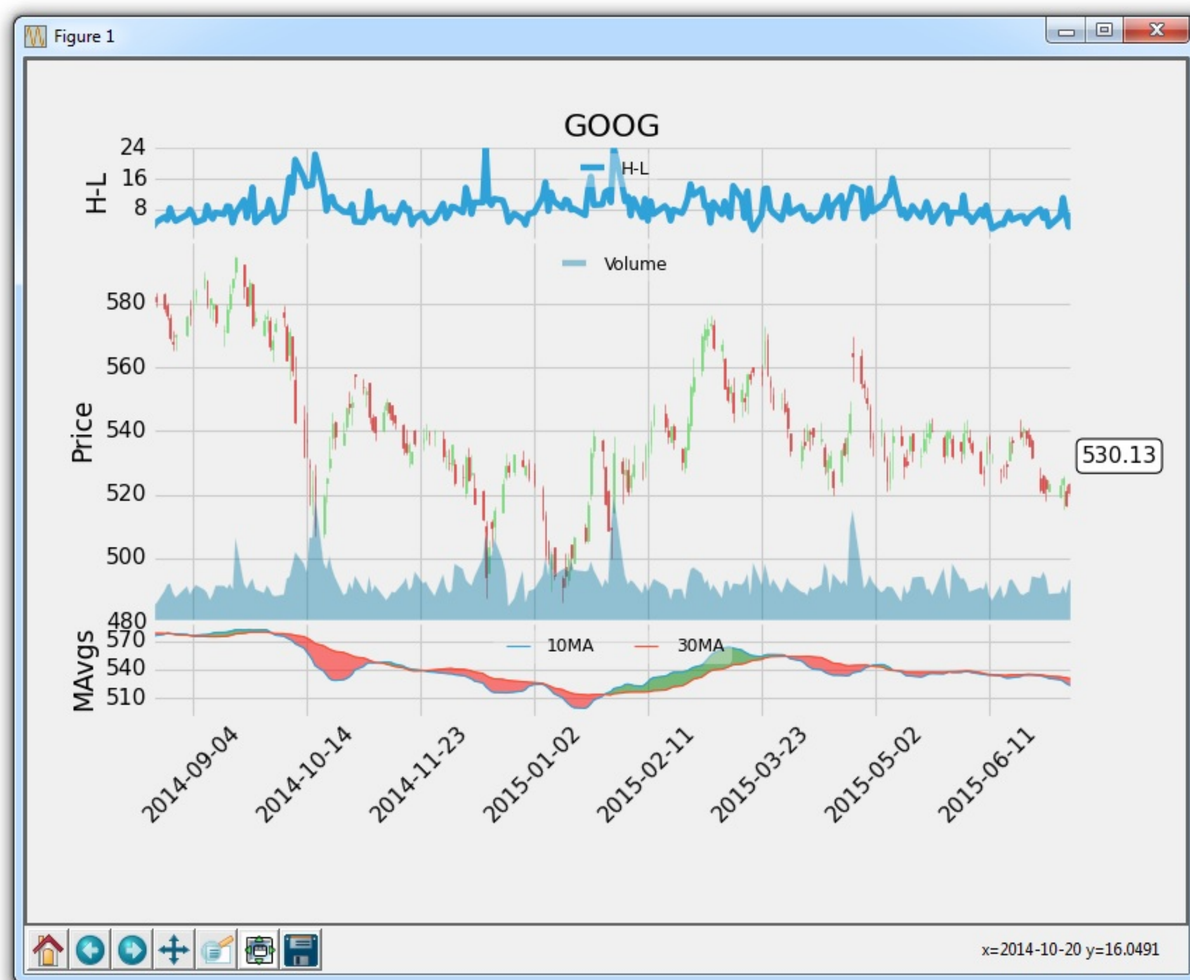
```
ax1.legend()
leg = ax1.legend(loc=9, ncol=2,prop={'size':11})
leg.get_frame().set_alpha(0.4)

ax2v.legend()
leg = ax2v.legend(loc=9, ncol=2,prop={'size':11})
leg.get_frame().set_alpha(0.4)

ax3.legend()
leg = ax3.legend(loc=9, ncol=2,prop={'size':11})
leg.get_frame().set_alpha(0.4)
```

所有的图例位于位置 9（上中间）。 有很多地方可放置图例，我们可以为参数传入不同的位置号码，来看看它们都位于哪里。 `ncol` 参数允许我们指定图例中的列数。 这里只有一列，如果图例中有 2 个项目，他们将堆叠在一列中。 最后，我们将尺寸规定为更小。 之后，我们对整个图例应用 `0.4` 的透明度。

现在我们的结果为：



完整的代码为：

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.finance import candlestick_ohlc
from matplotlib import style

import numpy as np
import urllib
import datetime as dt

style.use('fivethirtyeight')
print(plt.style.available)

print(plt.__file__)

MA1 = 10
MA2 = 30

def moving_average(values, window):
    weights = np.repeat(1.0, window)/window
    smas = np.convolve(values, weights, 'valid')
```

```
    return smas

def high_minus_low(highs, lows):
    return highs-lows


def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = mdates.strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


def graph_data(stock):

    fig = plt.figure(facecolor='#f0f0f0')
    ax1 = plt.subplot2grid((6,1), (0,0), rowspan=1, colspan=1)
    plt.title(stock)
    plt.ylabel('H-L')
    ax2 = plt.subplot2grid((6,1), (1,0), rowspan=4, colspan=1, s
harex=ax1)
    plt.ylabel('Price')
    ax2v = ax2.twinx()

    ax3 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1, s
harex=ax1)
    plt.ylabel('MAvgs')


    stock_price_url = 'http://chartapi.finance.yahoo.com/instrum
ent/1.0/'+stock+'/chartdata;type=quote;range=1y/csv'
    source_code = urllib.request.urlopen(stock_price_url).read()
.decode()
    stock_data = []
    split_source = source_code.split('\n')
    for line in split_source:
        split_line = line.split(',')
        if len(split_line) == 6:
            if 'values' not in line and 'labels' not in line:
                stock_data.append(line)


    date, closep, highp, lowp, openp, volume = np.loadtxt(stock_
data,
                                                          delimi
ter=',',
                                                          unpack=
True,
                                                          conver
ters={0: bytespdate2num('%Y%m%d')})

    x = 0
```

```
    y = len(date)
    ohlc = []

    while x < y:
        append_me = date[x], openp[x], highp[x], lowp[x], closep
[x], volume[x]
        ohlc.append(append_me)
        x+=1

    ma1 = moving_average(closep,MA1)
    ma2 = moving_average(closep,MA2)
    start = len(date[MA2-1:])

    h_l = list(map(high_minus_low, highp, lowp))


    ax1.plot_date(date[-start:],h_l[-start:],'-', label='H-L')
    ax1.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='lower'))


    candlestick_ohlc(ax2, ohlc[-start:], width=0.4, colorup='#77
d879', colordown='#db3f3f')



    ax2.yaxis.set_major_locator(mticker.MaxNLocator(nbins=7, pru
ne='upper'))
    ax2.grid(True)

    bbox_props = dict(boxstyle='round',fc='w', ec='k',lw=1)

    ax2.annotate(str(closep[-1]), (date[-1], closep[-1]),
                 xytext = (date[-1]+4, closep[-1]), bbox=bbox_pr
ops)

##    # Annotation example with arrow
##    ax2.annotate('Bad News!',(date[11],highp[11]),
##                 xytext=(0.8, 0.9), textcoords='axes fraction',

##                 arrowprops = dict(facecolor='grey',color='gre
y'))
##
##
##    # Font dict example
##    font_dict = {'family':'serif',
##                 'color':'darkred',
##                 'size':15}
##    # Hard coded text
##    ax2.text(date[10], closep[1],'Text Example', fontdict=font
_dict)

    ax2v.plot([],[], color='#0079a3', alpha=0.4, label='Volume')
```

```
    ax2v.fill_between(date[-start:],0, volume[-start:], facecolo
r='#0079a3', alpha=0.4)
    ax2v.axes.yaxis.set_ticklabels([])
    ax2v.grid(False)
    ax2v.set_ylim(0, 3*volume.max())



    ax3.plot(date[-start:], ma1[-start:], linewidth=1, label=(st
r(MA1)+'MA'))
    ax3.plot(date[-start:], ma2[-start:], linewidth=1, label=(st
r(MA2)+'MA'))

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] < ma2[-start:]),
                     facecolor='r', edgecolor='r', alpha=0.5)

    ax3.fill_between(date[-start:], ma2[-start:], ma1[-start:],
                     where=(ma1[-start:] > ma2[-start:]),
                     facecolor='g', edgecolor='g', alpha=0.5)

    ax3.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'
))
    ax3.xaxis.set_major_locator(mticker.MaxNLocator(10))
    ax3.yaxis.set_major_locator(mticker.MaxNLocator(nbins=4, pru
ne='upper'))

    for label in ax3.xaxis.get_ticklabels():
        label.set_rotation(45)



    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.setp(ax2.get_xticklabels(), visible=False)
    plt.subplots_adjust(left=0.11, bottom=0.24, right=0.90, top=
0.90, wspace=0.2, hspace=0)

    ax1.legend()
    leg = ax1.legend(loc=9, ncol=2,prop={'size':11})
    leg.get_frame().set_alpha(0.4)

    ax2v.legend()
    leg = ax2v.legend(loc=9, ncol=2,prop={'size':11})
    leg.get_frame().set_alpha(0.4)

    ax3.legend()
    leg = ax3.legend(loc=9, ncol=2,prop={'size':11})
    leg.get_frame().set_alpha(0.4)

    plt.show()
    fig.savefig('google.png', facecolor=fig.get_facecolor())
```

```
graph_data('GOOG')
```

现在我们可以看到图例，但也看到了图例下的任何信息。 还要注意额外函数 `fig.savefig` 。 这是自动保存图形的图像的方式。 我们还可以设置所保存的图形的前景色，使背景不是白色的，如我们的例子所示。

这就是目前为止，我想要显示的典型 Matplotlib 图表。 接下来，我们将涉及 `Basemap` ，它是一个 Matplotlib 扩展，用于绘制地理位置，然后我打算讲解 Matplotlib 中的 3D 图形。

第二十六章 Basemap 地理绘图

> 原文：Basemap Geographic Plotting with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们将涉及地理绘图模块`Basemap`。`Basemap`是 Matplotlib 的扩展。

为了使用`Basemap`，我们首先需要安装它。为了获得`Basemap`，你可以从这里获取：http://matplotlib.org/basemap/users/download.html，或者你可以访问 http://www.lfd.uci.edu/~gohlke/pythonlibs/。

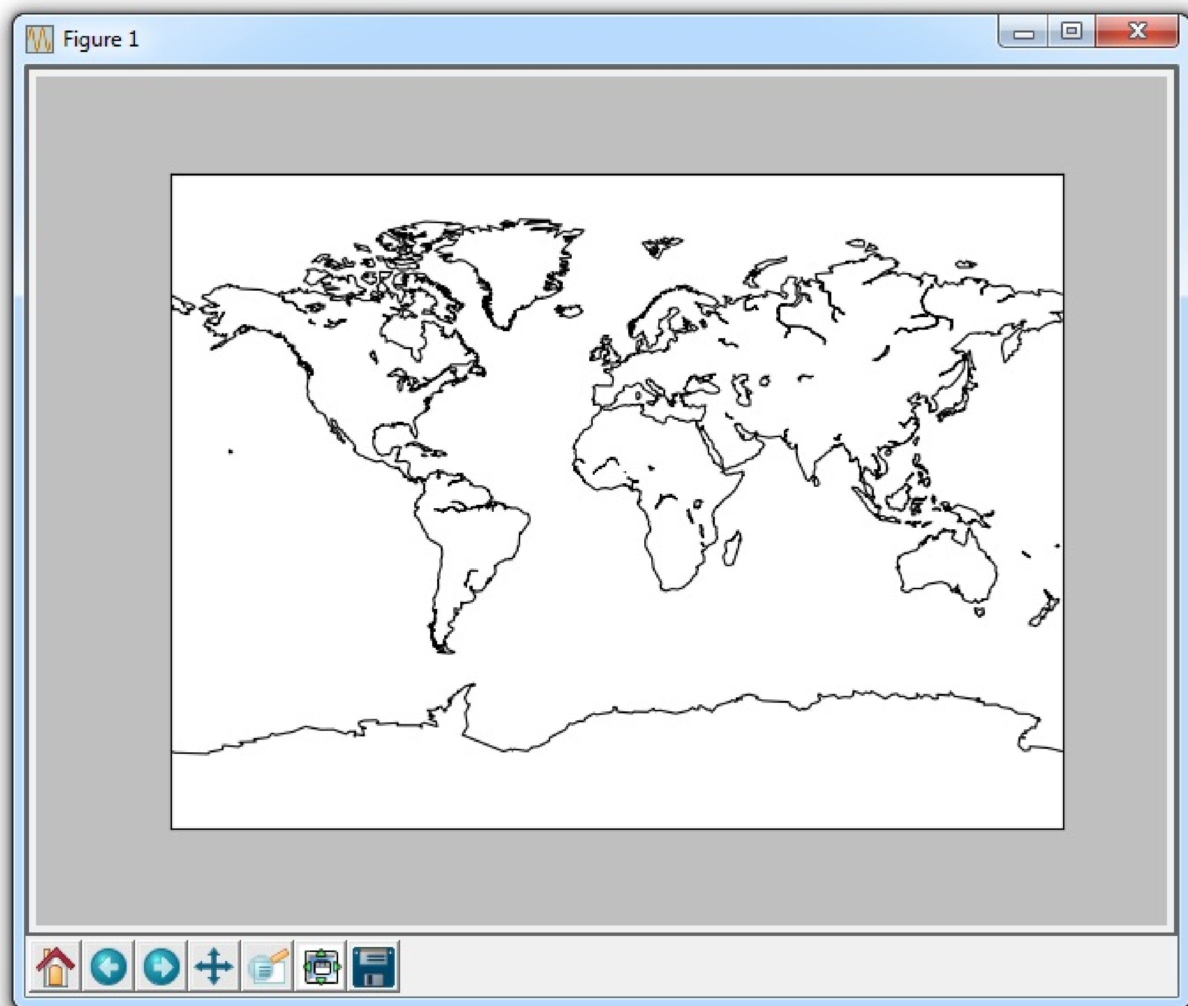
如果你在安装`Basemap`时遇到问题，请查看`pip`安装教程。

一旦你安装了`Basemap`，你就可以创建地图了。首先，让我们投影一个简单的地图。为此，我们需要导入`Basemap`，`pyplot`，创建投影，至少绘制某种轮廓或数据，然后我们可以显示图形。

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

m = Basemap(projection='mill')
m.drawcoastlines()
plt.show()
```

上面的代码结果如下：



这是使用 Miller 投影完成的，这只是许多 `Basemap` 投影选项之一。

第二十七章 Basemap 自定义

> 原文：Basemap Customization with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这篇 Matplotlib 教程中，我们继续使用 `Basemap` 地理绘图扩展。 我们将展示一些我们可用的自定义选项。

首先，从上一个教程中获取我们的起始代码：

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

m = Basemap(projection='mill')
m.drawcoastlines()
plt.show()
```

我们可以从放大到特定区域来开始：

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

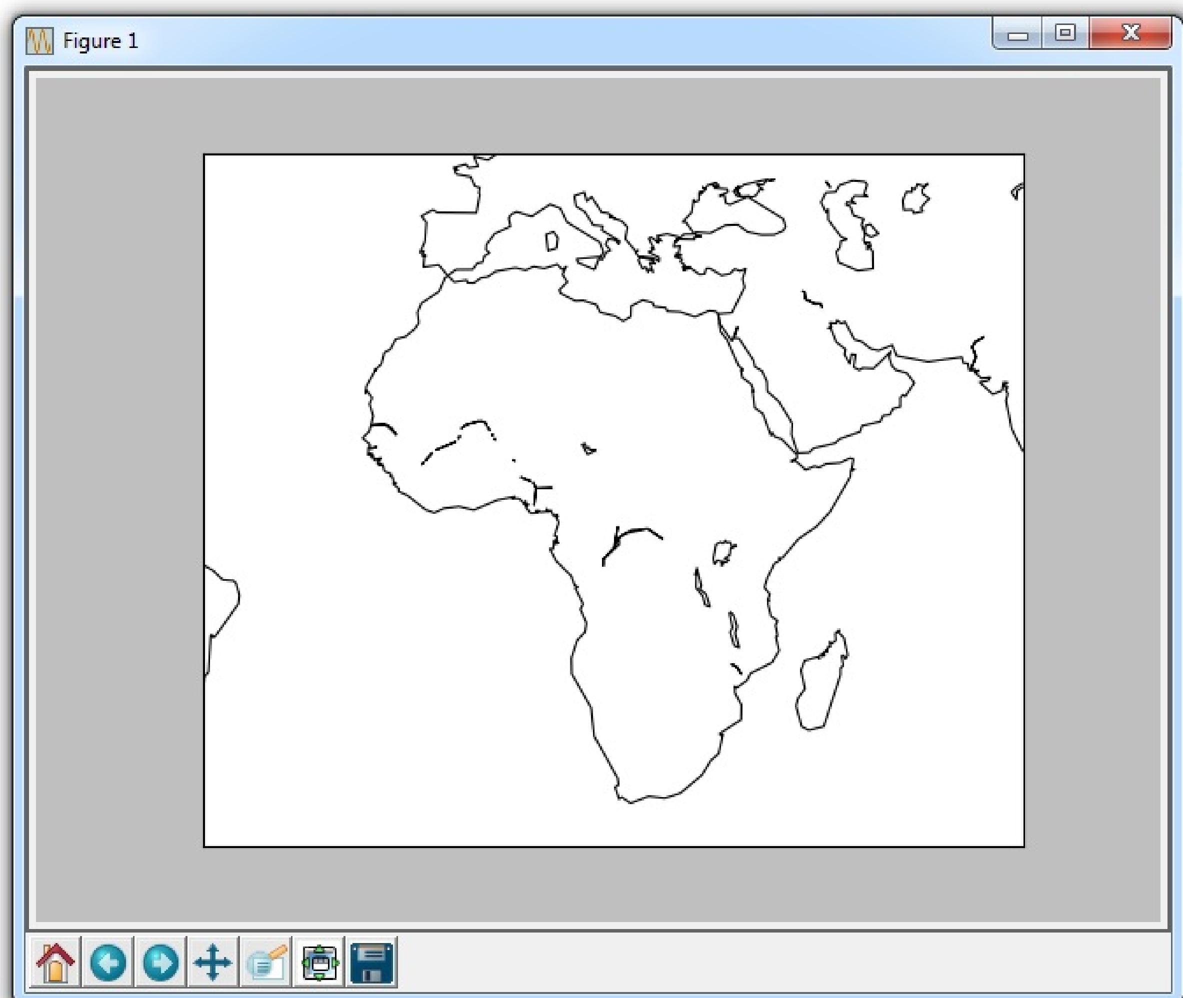
m = Basemap(projection='mill',
            llcrnrlat = -40,
            llcrnrlon = -40,
            urcrnrlat = 50,
            urcrnrlon = 75)
m.drawcoastlines()
plt.show()
```

这里的参数是：

- `llcrnrlat` - 左下角的纬度
- `llcrnrlon` - 左下角的经度
- `urcrnrlat` - 右上角的纬度
- `urcrnrlon` - 右上角的经度

此外，坐标需要转换，其中西经和南纬坐标是负值，北纬和东经坐标是正值。

使用这些坐标，`Basemap` 会选择它们之间的区域。



下面，我们要使用一些东西，类似：

```
m.drawcountries(linewidth=2)
```

这会画出国家，并使用线宽为 2 的线条生成分界线。

另一个选项是：

```
m.drawstates(color='b')
```

这会用蓝色线条画出州。

你也可以执行：

```
m.drawcounties(color='darkred')
```

这会画出国家。

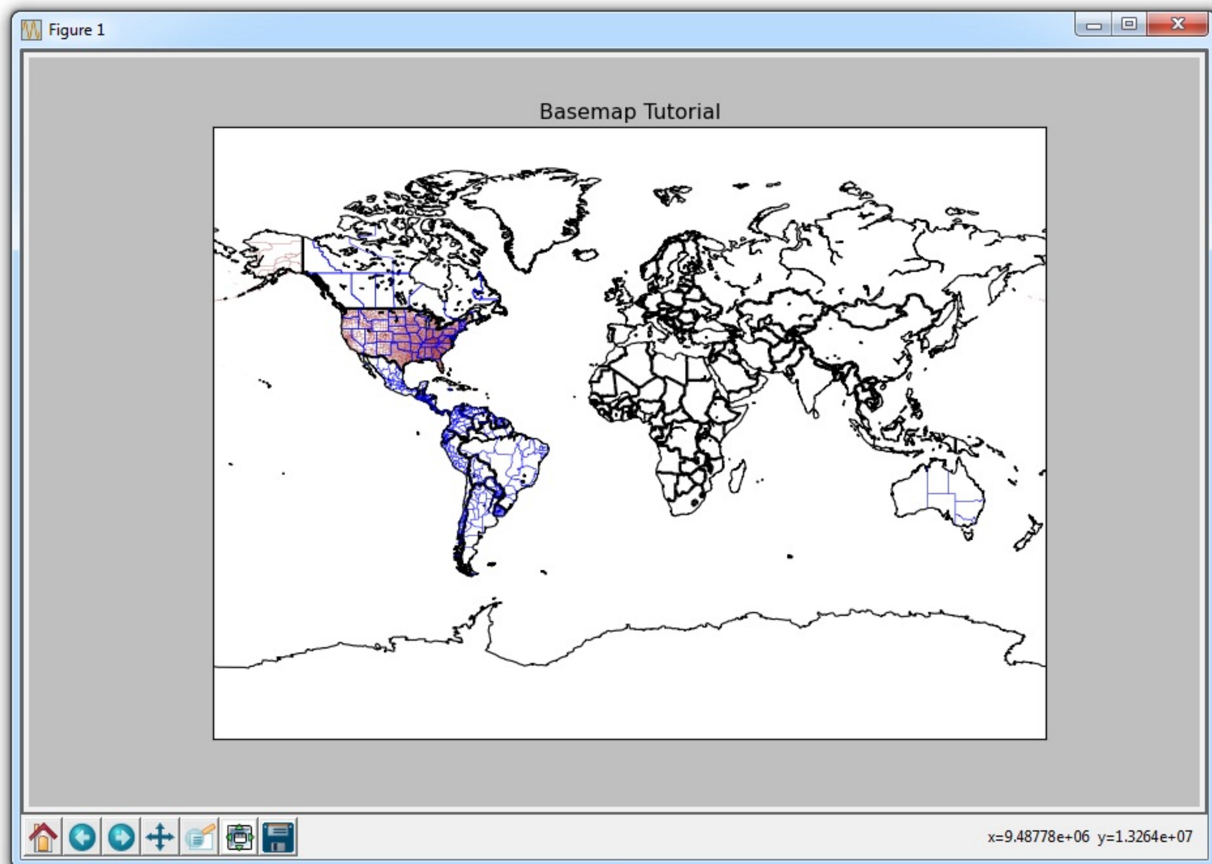
所以，我们的代码是：

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

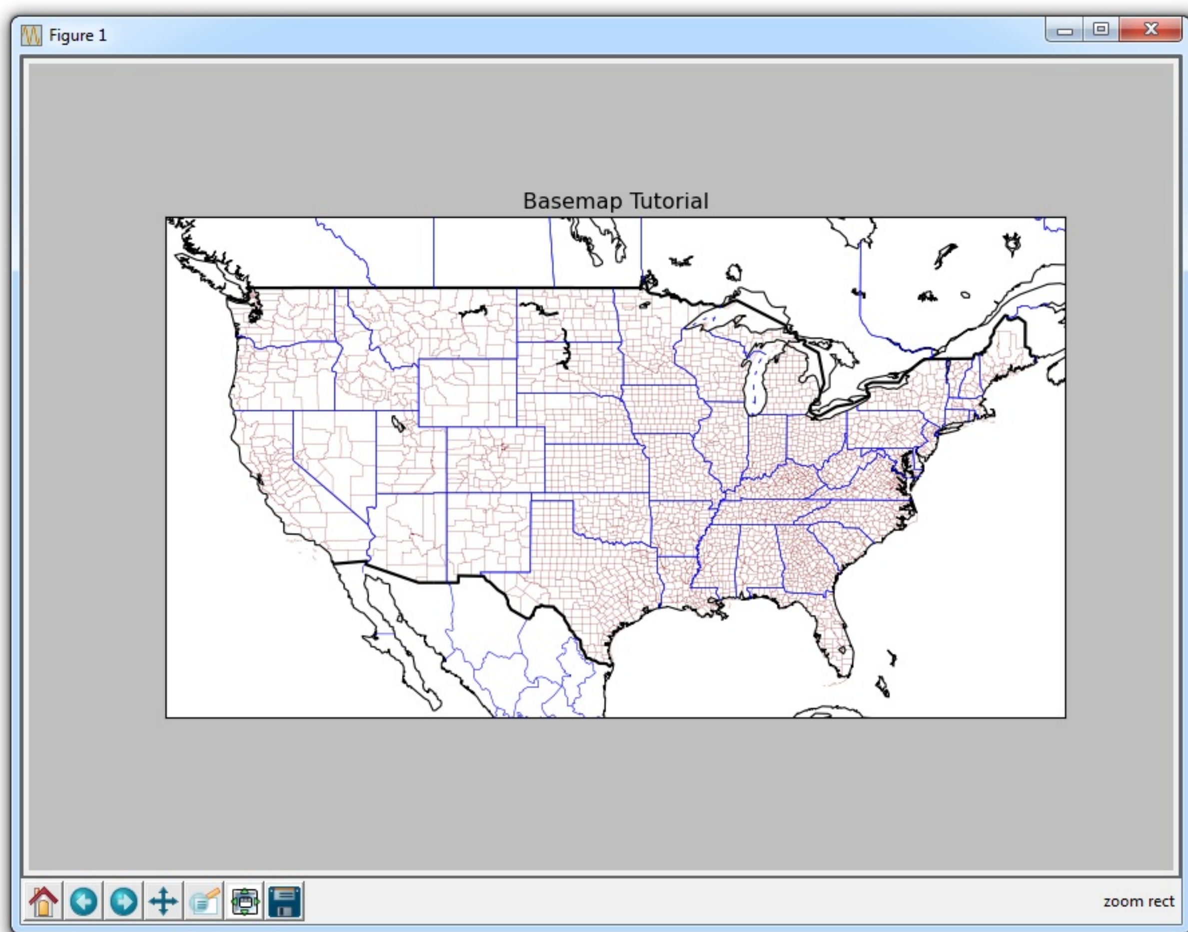
m = Basemap(projection='mill',
            llcrnrlat = -90,
            llcrnrlon = -180,
            urcrnrlat = 90,
            urcrnrlon = 180)

m.drawcoastlines()
m.drawcountries(linewidth=2)
m.drawstates(color='b')
m.drawcounties(color='darkred')

plt.title('Basemap Tutorial')
plt.show()
```



很难说，但我们定义了美国的区县的线条。 我们可以使用放大镜放大 `Basemap` 图形，就像其他图形那样，会生成：



另一个有用的选项是 `Basemap` 调用中的『分辨率』选项。

```
m = Basemap(projection='mill',
            llcrnrlat = -90,
            llcrnrlon = -180,
            urcrnrlat = 90,
            urcrnrlon = 180,
            resolution='l')
```

分辨率的选项为：

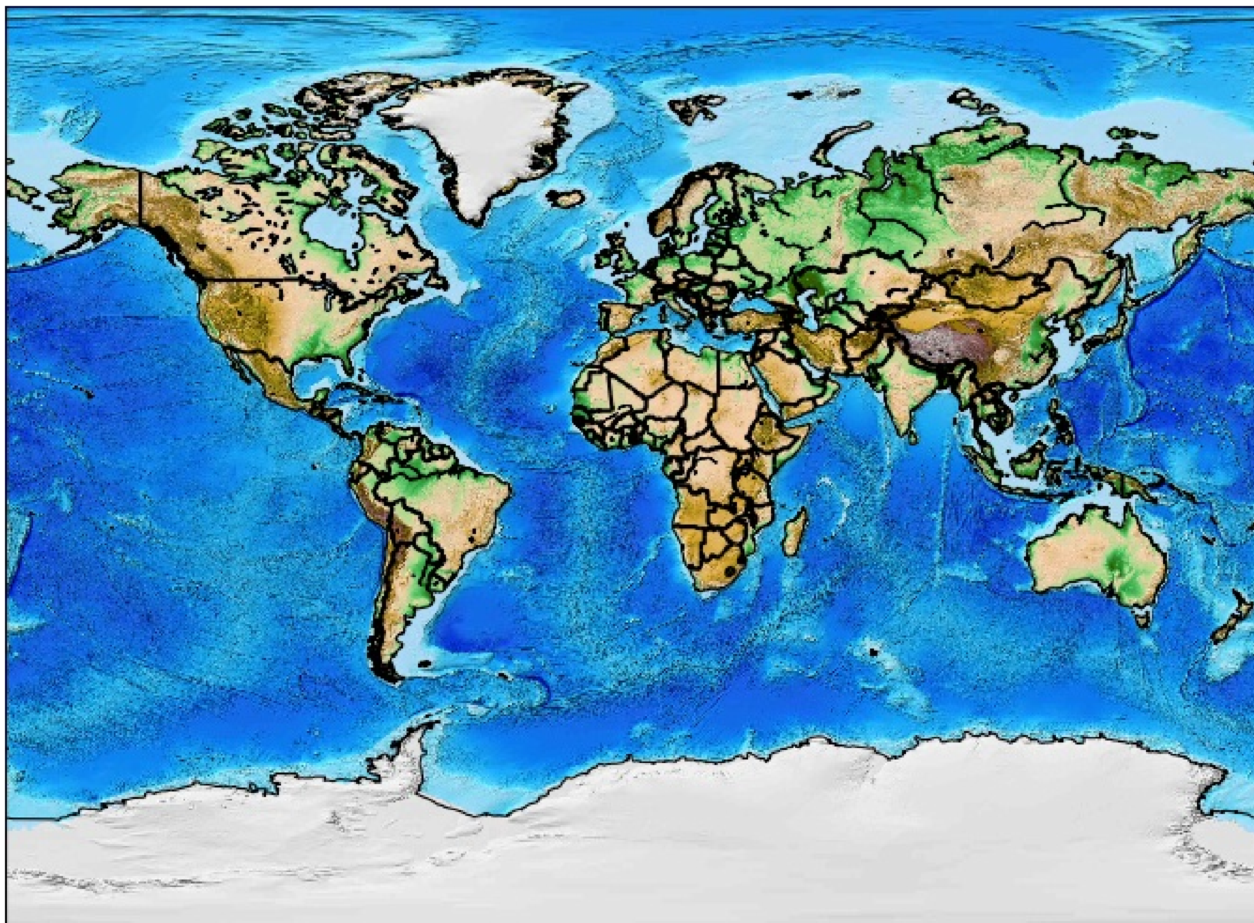
- `c` - 粗糙
- `l` - 低
- `h` - 高
- `f` - 完整

对于更高的分辨率，你应该放大到很大，否则这可能只是浪费。

另一个选项是使用 `etopo()` 绘制地形，如：

```
m.etopo()
```

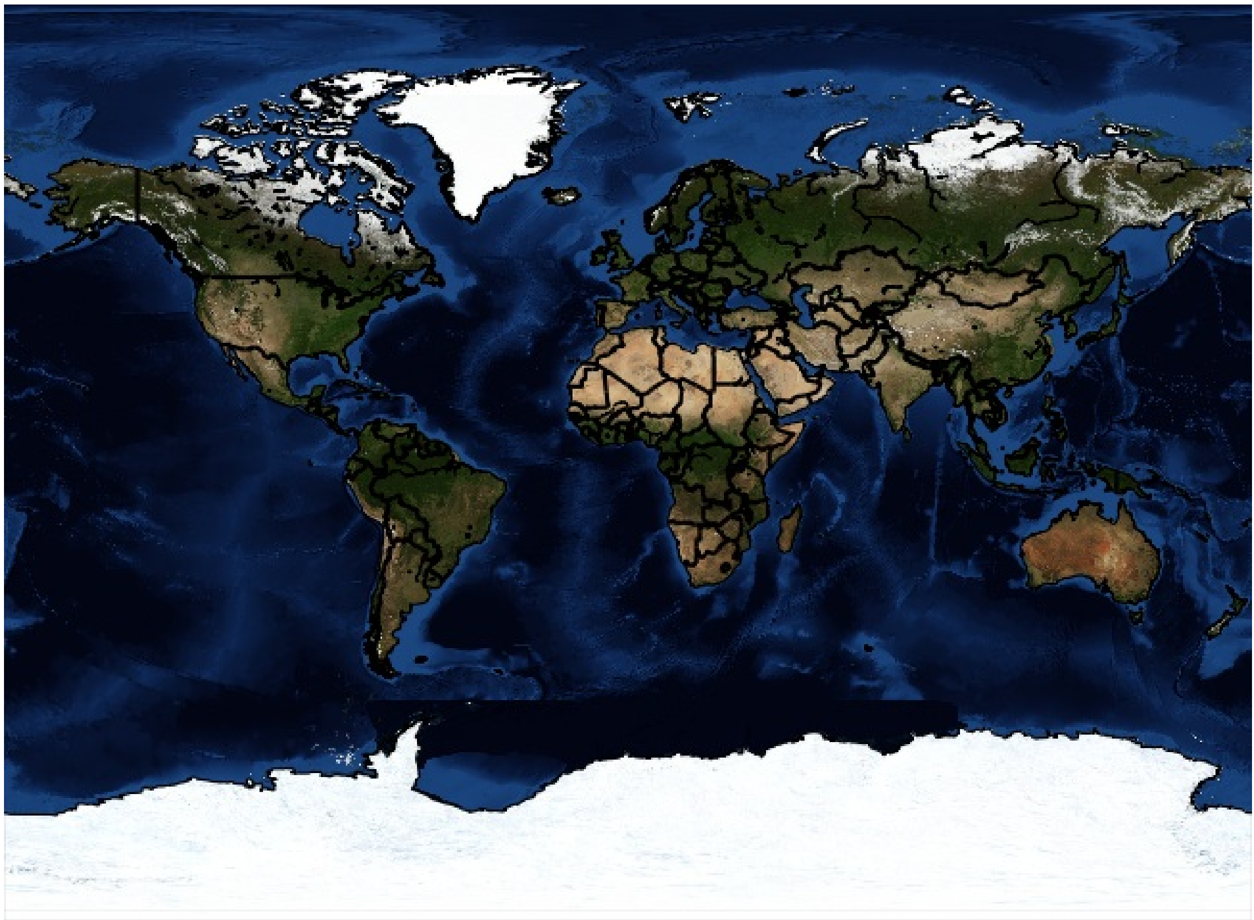
使用 `drawcountries` 方法绘制此图形会生成：



最后，有一个蓝色的大理石版本，你可以调用：

```
m.bluemarble()
```

会生成：



目前为止的代码：

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

m = Basemap(projection='mill',
            llcrnrlat = -90,
            llcrnrlon = -180,
            urcrnrlat = 90,
            urcrnrlon = 180,
            resolution='l')

m.drawcoastlines()
m.drawcountries(linewidth=2)
##m.drawstates(color='b')
##m.drawcounties(color='darkred')
#m.fillcontinents()
#m.etopo()
m.bluemarble()

plt.title('Basemap Tutorial')
plt.show()
`
```

第二十八章 在 Basemap 中绘制坐标

> 原文：Plotting Coordinates in Basemap with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

欢迎阅读另一个 Matplotlib Basemap 教程。 在本教程中，我们将介绍如何绘制单个坐标，以及如何在地理区域中连接这些坐标。

首先，我们将从一些基本的起始数据开始：

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

m = Basemap(projection='mill',
            llcrnrlat = 25,
            llcrnrlon = -130,
            urcrnrlat = 50,
            urcrnrlon = -60,
            resolution='l')

m.drawcoastlines()
m.drawcountries(linewidth=2)
m.drawstates(color='b')
```

接下来，我们可以绘制坐标，从获得它们的实际坐标开始。 记住，南纬和西经坐标需要转换为负值。 例如，纽约市是北纬 `40.7127` 西经 `74.0059` 。 我们可以在我们的程序中定义这些坐标，如：

```
NYClat, NYClon = 40.7127, -74.0059
```

之后我们将这些转换为要绘制的 `x` 和 `y` 坐标。

```
xpt, ypt = m(NYClon, NYClat)
```

注意这里，我们现在已经将坐标顺序翻转为 `lon, lat` （纬度，经度）。 坐标通常以 `lat, lon` 顺序给出。 然而，在图形中， `lat, long` 转换为 `y, x` ，我们显然不需要。 在某些时候，你必须翻转它们。 不要忘记这部分！

最后，我们可以绘制如下的坐标：

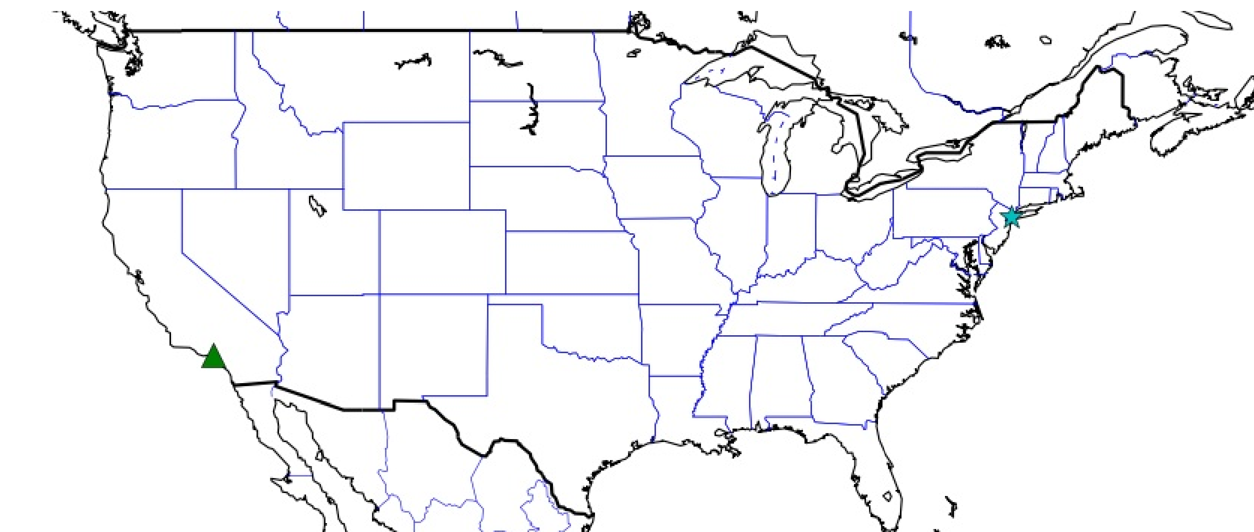
```
m.plot(xpt, ypt, 'c*', markersize=15)
```

这个图表上有一个青色的星，大小为 15。更多标记类型请参阅：Matplotlib 标记文档。

接下来，让我们再画一个位置，洛杉矶，加利福尼亚：

```
LAlat, LAlon = 34.05, -118.25
xpt, ypt = m(LAlon, LAlat)
m.plot(xpt, ypt, 'g^', markersize=15)
```

这次我们画出一个绿色三角，执行代码会生成：



如果我们想连接这些图块怎么办？原来，我们可以像其它 Matplotlib 图表那样实现它。

首先，我们将那些 `xpt` 和 `ypt` 坐标保存到列表，类似这样的东西：

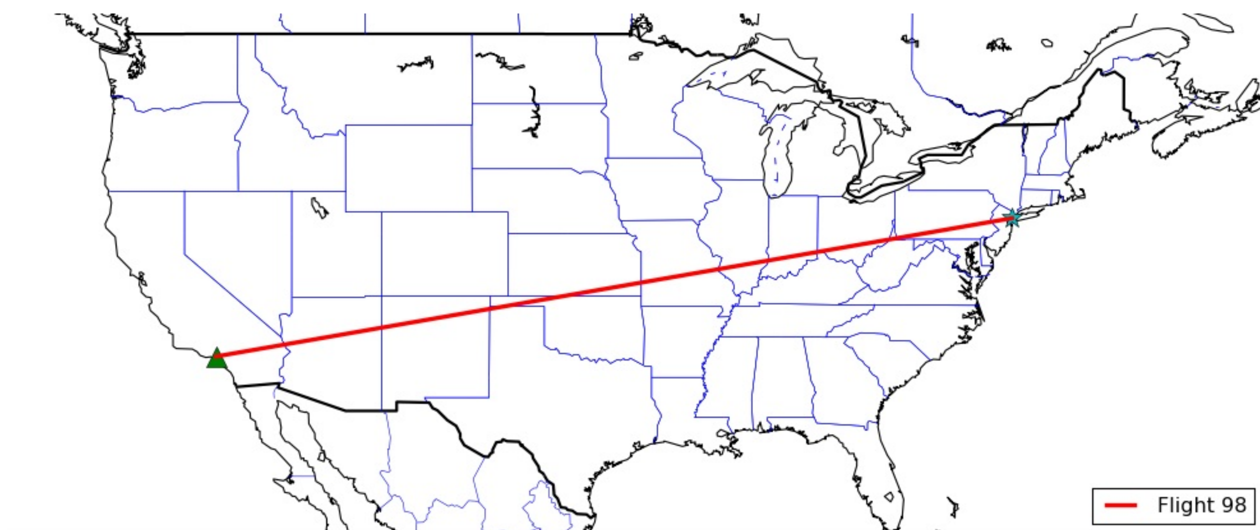
```
xs = []
ys = []

NYClat, NYClon = 40.7127, -74.0059
xpt, ypt = m(NYClon, NYClat)
xs.append(xpt)
ys.append(ypt)
m.plot(xpt, ypt, 'c*', markersize=15)

LAlat, LAlon = 34.05, -118.25
xpt, ypt = m(LAlon, LAlat)
xs.append(xpt)
ys.append(ypt)
m.plot(xpt, ypt, 'g^', markersize=15)

m.plot(xs, ys, color='r', linewidth=3, label='Flight 98')
```

会生成：



太棒了。有时我们需要以圆弧连接图上的两个坐标。如何实现呢？

```
m.drawgreatcircle(NYClon, NYClat, LAlon, LAlat, color='c', linewidth=3, label='Arc')
```

我们的完整代码为：

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

m = Basemap(projection='mill',
            llcrnrlat = 25,
            llcrnrlon = -130,
            urcrnrlat = 50,
            urcrnrlon = -60,
            resolution='l')

m.drawcoastlines()
m.drawcountries(linewidth=2)
m.drawstates(color='b')
#m.drawcounties(color='darkred')
#m.fillcontinents()
#m.etopo()
#m.bluemarble()

xs = []
ys = []

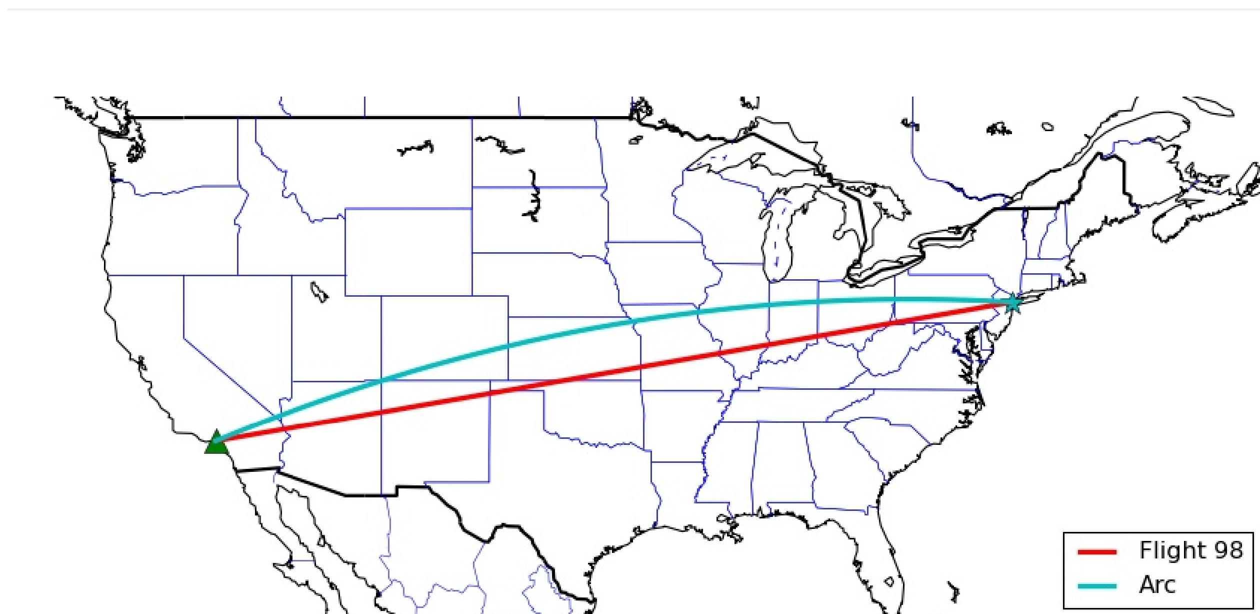
NYClat, NYClon = 40.7127, -74.0059
xpt, ypt = m(NYClon, NYClat)
xs.append(xpt)
ys.append(ypt)
m.plot(xpt, ypt, 'c*', markersize=15)

LAlat, LAlon = 34.05, -118.25
xpt, ypt = m(LAlon, LAlat)
xs.append(xpt)
ys.append(ypt)
m.plot(xpt, ypt, 'g^', markersize=15)

m.plot(xs, ys, color='r', linewidth=3, label='Flight 98')
m.drawgreatcircle(NYClon, NYClat, LAlon, LAlat, color='c', linew
idth=3, label='Arc')


plt.legend(loc=4)
plt.title('Basemap Tutorial')
plt.show()
```

结果为：



这就是 `Basemap` 的全部了，下一章关于 Matplotlib 的 3D 绘图。

第二十九章 3D 绘图

> 原文：3D graphs with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

您好，欢迎阅读 Matplotlib 教程中的 3D 绘图。 Matplotlib 已经内置了三维图形，所以我们不需要再下载任何东西。 首先，我们需要引入一些完整的模块：

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
```

使用 `axes3d` 是因为它需要不同种类的轴域，以便在三维中实际绘制一些东西。下面：

```
fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')
```

在这里，我们像通常一样定义图形，然后我们将 `ax1` 定义为通常的子图，只是这次使用 3D 投影。 我们需要这样做，以便提醒 Matplotlib 我们要提供三维数据。

现在让我们创建一些 3D 数据：

```
x = [1,2,3,4,5,6,7,8,9,10]
y = [5,6,7,8,2,5,6,3,7,2]
z = [1,2,6,3,2,7,3,3,7,2]
```

接下来，我们绘制它。 首先，让我们展示一个简单的线框示例：

```
ax1.plot_wireframe(x,y,z)
```

最后：

```
ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt.show()
```

我们完整的代码是：

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import style

style.use('fivethirtyeight')

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')

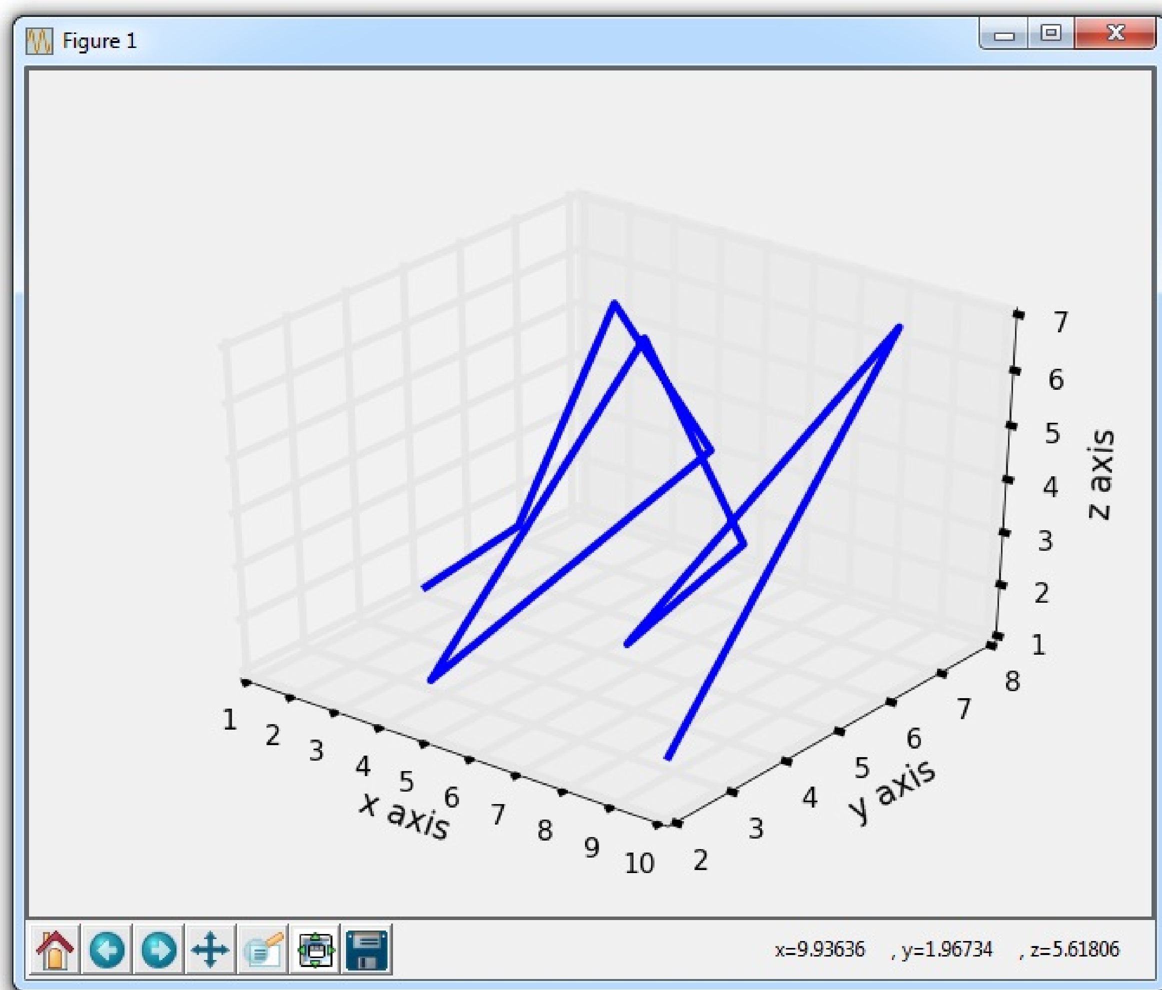
x = [1,2,3,4,5,6,7,8,9,10]
y = [5,6,7,8,2,5,6,3,7,2]
z = [1,2,6,3,2,7,3,3,7,2]

ax1.plot_wireframe(x,y,z)

ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt.show()
```

结果为（包括所用的样式）：



这些 3D 图形可以进行交互。首先，您可以使用鼠标左键单击并拖动来移动图形。您还可以使用鼠标右键单击并拖动来放大或缩小。

第三十章 3D 散点图

> 原文：3D Scatter Plot with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

欢迎阅读另一个 3D Matplotlib 教程，会涉及如何绘制三维散点图。

绘制 3D 散点图非常类似于通常的散点图以及 3D 线框图。

一个简单示例：

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import style

style.use('ggplot')

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')

x = [1,2,3,4,5,6,7,8,9,10]
y = [5,6,7,8,2,5,6,3,7,2]
z = [1,2,6,3,2,7,3,3,7,2]

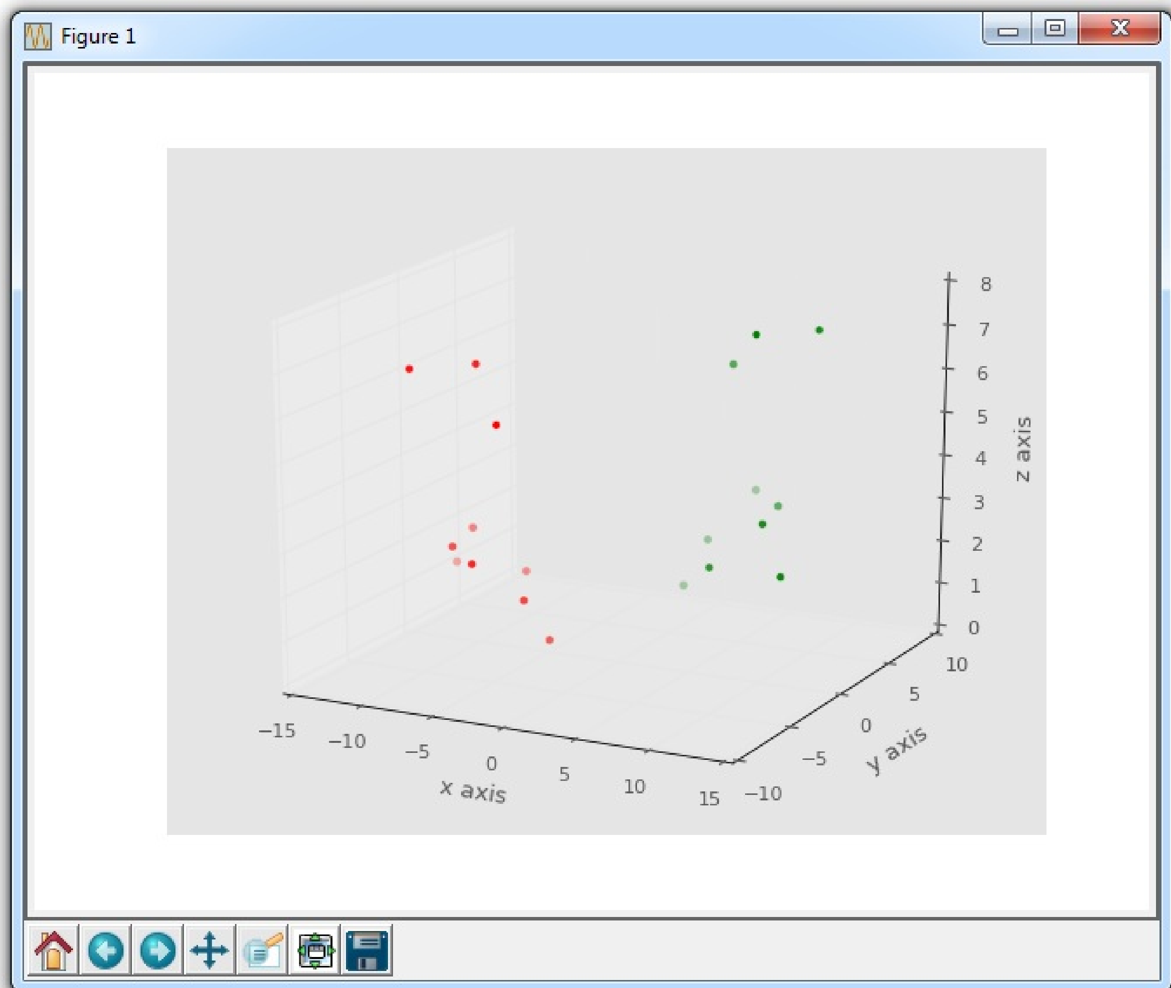
x2 = [-1,-2,-3,-4,-5,-6,-7,-8,-9,-10]
y2 = [-5,-6,-7,-8,-2,-5,-6,-3,-7,-2]
z2 = [1,2,6,3,2,7,3,3,7,2]

ax1.scatter(x, y, z, c='g', marker='o')
ax1.scatter(x2, y2, z2, c ='r', marker='o')

ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt.show()
```

结果为：



要记住你可以修改这些绘图的大小和标记，就像通常的散点图那样。

第三十一章 3D 条形图

> 原文：3D Bar Chart with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

在这个 Matplotlib 教程中，我们要介绍 3D 条形图。 3D 条形图是非常独特的，因为它允许我们绘制多于 3 个维度。 不，你不能超过第三个维度来绘制，但你可以绘制多于 3 个维度。

对于条形图，你需要拥有条形的起点，条形的高度和宽度。 但对于 3D 条形图，你还有另一个选项，就是条形的深度。 大多数情况下，条形图从轴上的条形平面开始，但是你也可以通过打破此约束来添加另一个维度。 然而，我们会让它非常简单：

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
style.use('ggplot')

fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')

x3 = [1,2,3,4,5,6,7,8,9,10]
y3 = [5,6,7,8,2,5,6,3,7,2]
z3 = np.zeros(10)

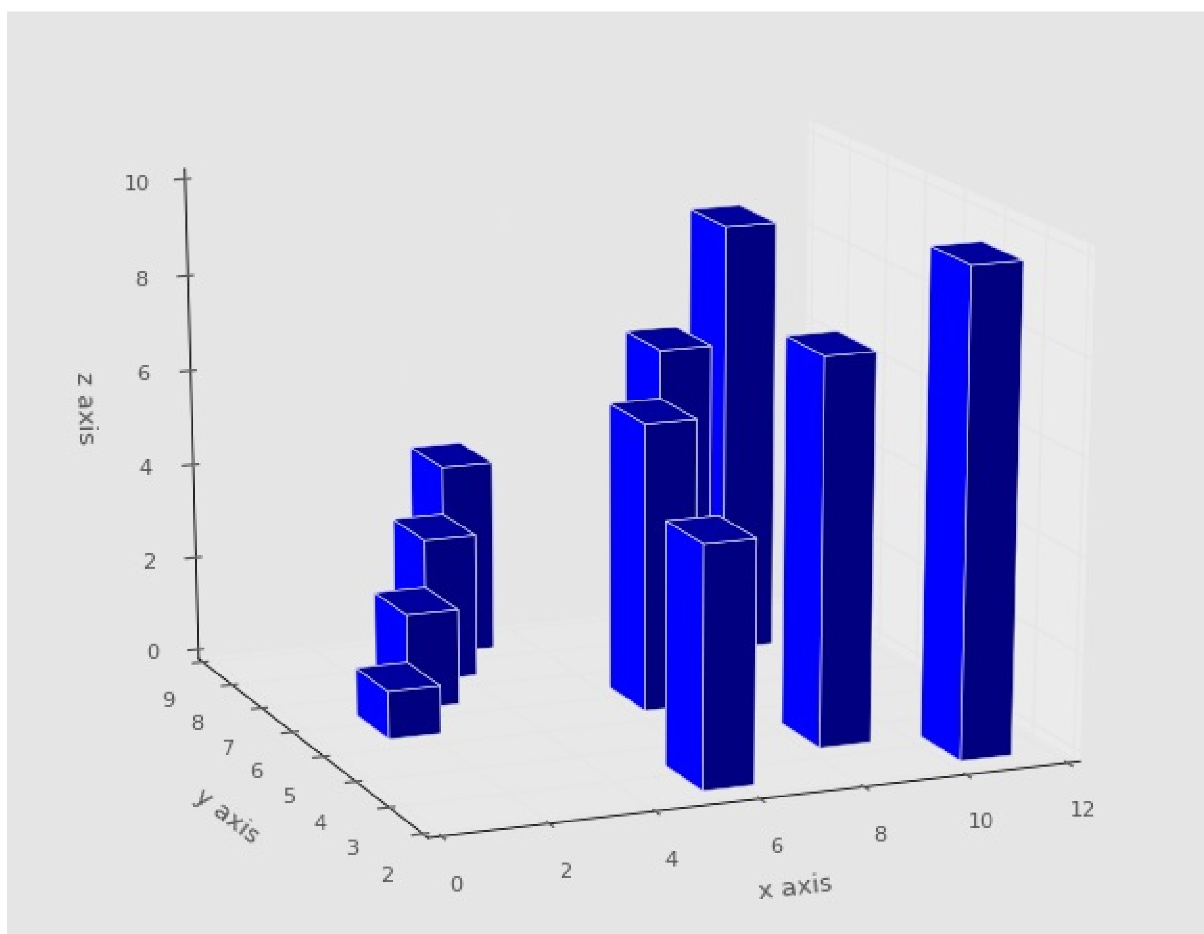
dx = np.ones(10)
dy = np.ones(10)
dz = [1,2,3,4,5,6,7,8,9,10]

ax1.bar3d(x3, y3, z3, dx, dy, dz)


ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt.show()
```

注意这里，我们必须定义 `x` 、 `y` 和 `z` ，然后是 3 个维度的宽度、高度和深度。 这会生成：


10
8
6
4
2
0
z axis
9
8
7
6
5
4
3
2
y axis
0
2
4
6
8
10
12
x axis

第三十二章 总结

> 原文：Conclusion with Matplotlib
>
> 译者：飞龙
>
> 协议：CC BY-NC-SA 4.0

欢迎阅读最后的 Matplotlib 教程。在这里我们将整理整个系列，并显示一个稍微更复杂的 3D 线框图：

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
style.use('ggplot')

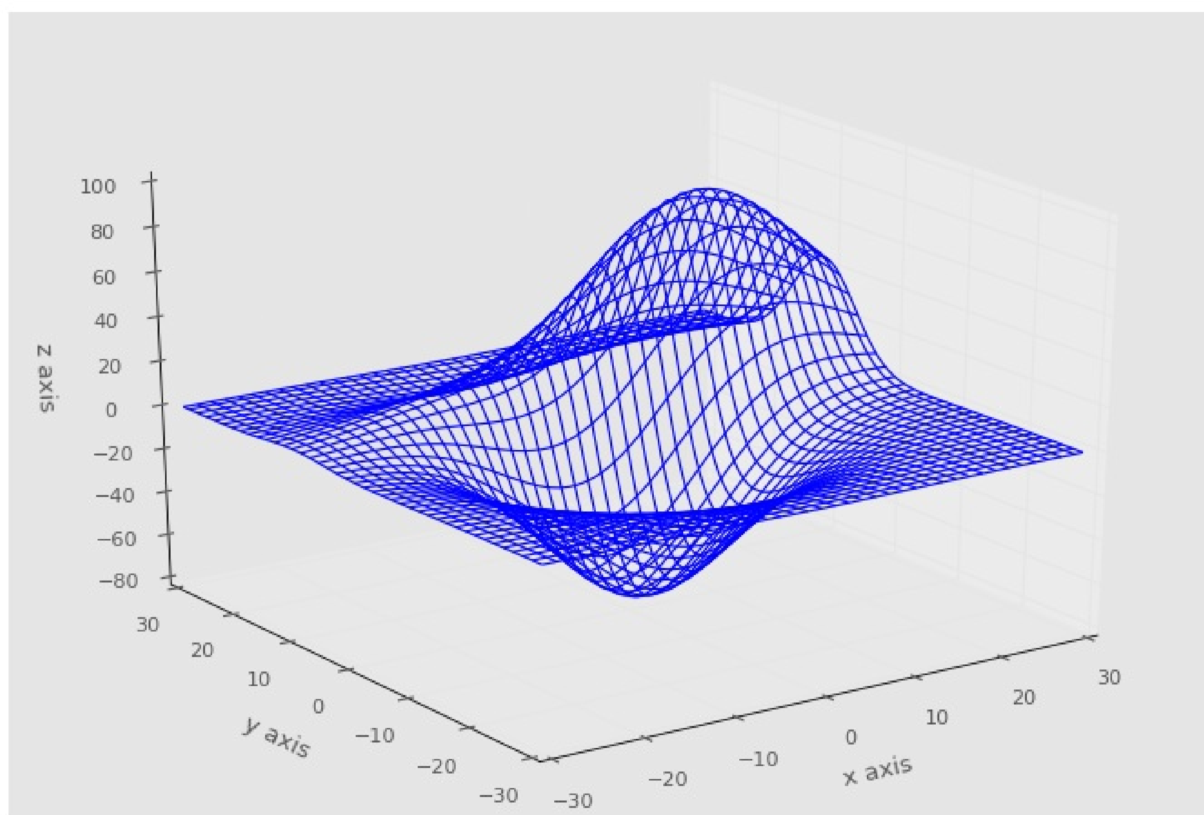
fig = plt.figure()
ax1 = fig.add_subplot(111, projection='3d')

x, y, z = axes3d.get_test_data()

print(axes3d.__file__)
ax1.plot_wireframe(x,y,z, rstride = 3, cstride = 3)

ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

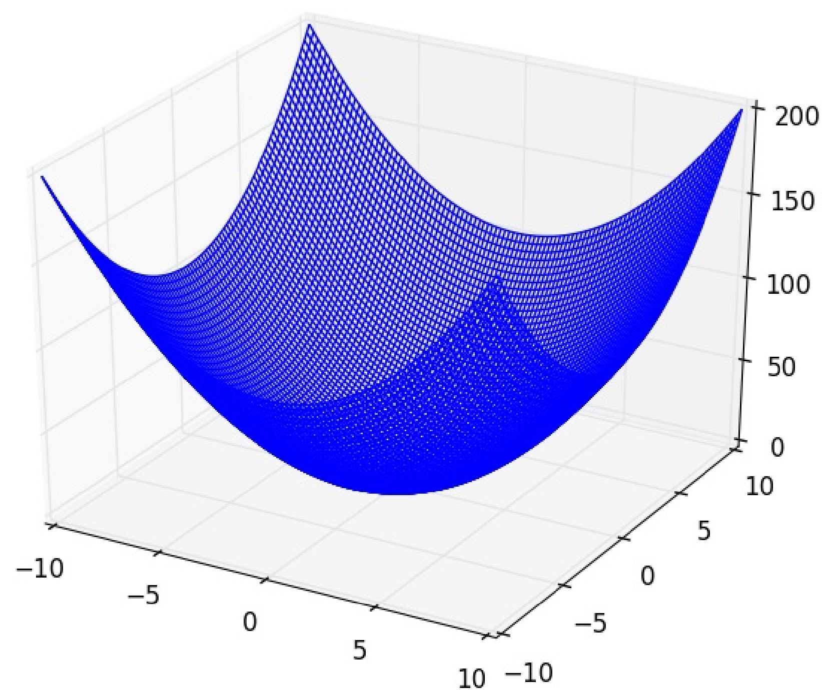
plt.show()
```



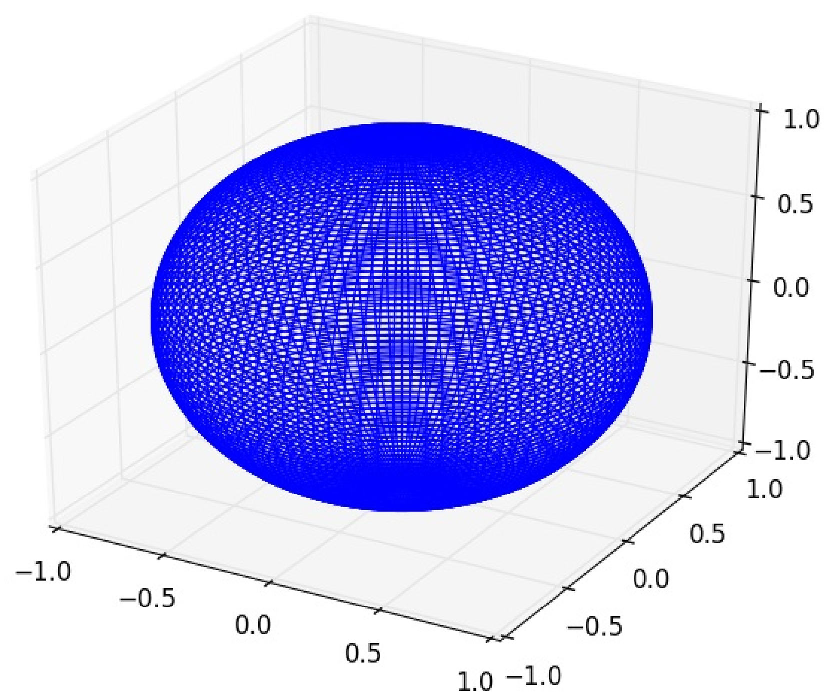
如果你从一开始就关注这个教程的话，那么你已经学会了 Matplotlib 提供的大部分内容。 你可能不相信，但Matplotlib 仍然可以做很多其他的事情！ 请继续学习，你可以随时访问 Matplotlib.org，并查看示例和图库页面。

如果你发现自己大量使用 Matplotlib，请考虑捐助给 John Hunter Memorial 基金。

> 注：空间曲面的画法
>
> ```
> # 二次抛物面 z = x^2 + y^2
> x = np.linspace(-10, 10, 101)
> y = x
> x, y = np.meshgrid(x, y)
> z = x ** 2 + y ** 2
> ax = plot.subplot(111, projection='3d')
> ax.plot_wireframe(x, y, z)
> plot.show()
> ```




```
# 半径为 1 的球
t = np.linspace(0, np.pi * 2, 100)
s = np.linspace(0, np.pi, 100)
t, s = np.meshgrid(t, s)
x = np.cos(t) * np.sin(s)
y = np.sin(t) * np.sin(s)
z = np.cos(s)
ax = plot.subplot(111, projection='3d')
ax.plot_wireframe(x, y, z)
plot.show()
```


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
−1.0
−0.5
0.0
0.5
1.0